TRANSIMS:
TRansportation ANalysis SIMulation System

Version: TRANSIMS –2.1

VOLUME TWO—NETWORKS AND VEHICLES

08 February 2002

LA-UR-00-1724

Los Alamos
NATIONAL LABORATORY

VOLUME TWO—NETWORKS AND VEHICLES

08 February 2002

The following individuals contributed to this document:

C. L. Barrett*
R. J. Beckman*
K. P. Berkbigler*
K. R. Bisset*
B. W. Bush*
K. Campbell*
S. Eubank*
K. M. Henson*
J. M. Hurford*
D. A. Kubicek*
M. V. Marathe*
P. R. Romero*
J. P. Smith*
L. L. Smith*
P. L. Speckman**
P. E. Stretz*
G. L. Thayer*
E. Van Eeckhout*
M. D. Williams*

* Los Alamos National Laboratory, Los Alamos, NM 87545
** National Institute of Statistical Sciences, Research Triangle Park, NC

Acknowledgments

This work was supported by the U. S. Department of Transportation (Assistant Secretary for Transportation Policy, Federal Highway Administration, Federal Transit Administration), the U.S. Environmental Protection Agency, and the U.S. Department of Energy as part of the Travel Model Improvement Program.

Volume Two: Contents

Chapter Ten: Figures

Chapter Ten: Tables

1. Network Files

The TRANSIMS Network representation provides detailed information about streets, intersections, signals, and transit in a road network. This section discusses the concepts involved in describing a road network and the TRANSIMS data table formats.

1.1 File Format

This section specifies the formats for the 18 data tables required to describe a TRANSIMS road network. Table 1 shows how the tables depend on one another. The units of measurement are SI units (i.e., distances in meters, time in seconds, etc.). Geographic coordinates are specified in the Universal Transverse Mercator (UTM) Grid System.

Table 1. Interdependencies between network tables.

Table	Tables on which it depends
Link	Node
Speed	Node, Link, Pocket Lane
Pocket Lane	Node, Link
Lane Use	Node, Link, Pocket Lane
Parking	Node, Link
Barrier	Node, Link, Pocket Lane
Transit Stop	Node, Link
Lane Connectivity	Node, Link, Pocket Lane
Turn Prohibition	Node, Link, Pocket Lane
Unsignalized Node	Node, Link, Pocket Lane
Signalized Node	Node, Timing Plan
Phasing Plan	Node, Link, Pocket Lane, Timing Plan
Detector	Node, Link, Pocket Lane
Signal Coordinator	Node, Signalized Node
Activity Location	Node, Link
Process Link	Parking, Transit Stop, Activity Location
Study Area Link	Link

The TRANSIMS software architecture allows for the inclusion of additional columns desired by an analyst, so the specification below gives only the required columns. The format for data files is ASCII, with columns delimited by tab characters; records are terminated by a new-line character (i.e., International Standards Organization ((ISO) format). The first line of the file must contain the field names (i.e., column headings) delimited by tab characters.

1.1.1 Node Table

Table 2 specifies the format for the node table. To validate a node table, verify the following:

- The field names and types are correct.
- The data values are in the legal ranges.
- The IDs are unique.
- No nodes have the same easting, northing, and elevation. Also acceptable are nodes with the same easting and northing, but different elevations.

Table 2. Node table format.

Column Name	Description	Allowed Values
`ID`	The ID number of the node.	integer: 1 through 2,147,483,647
`EASTING`	The x-coordinate of the node (in meters, UTM grid system).	floating-point number
`NORTHING`	The y-coordinate of the node (in meters, UTM grid system).	floating-point number
`ELEVATION`	The z-coordinate of the node (in meters, UTM grid system).	floating-point number
`NOTES`	A character string used for data quality annotations; free format (may be blank).	255 characters

1.1.2 Link Table

Table 3 specifies the format for the link table. To validate a link table, verify the following:

- The field names and types are correct.
- The data values are in the legal ranges.
- The IDs are unique.
- The nodes at the endpoints exist.
- There are different nodes at the endpoints.
- There are permanent lanes in at least one direction.
- There is at least one permanent lane in every direction that there is a pocket lane.
- The length of the link is at least as great as the distance between its endpoints.
- The length of the link is not far greater (e.g., 50% more) than the distance between its endpoints.
- The length of the link is not exceedingly small. (The Traffic Microsimulator may have difficulty simulating successive links that are less than 50 meters long.)

- The sum of the setback lengths is less than the length of the link.
- All nodes have at least one incoming and one outgoing link.
- At least some types of vehicles are allowed on the link.
- The functional classes of all the links connected to a node are consistent: Table 4 lists functional classes for links. First, divide the TRANSIMS functional classes into three categories:

  1) restricted—Freeway, Expressway
  2) surface—Primary Arterial, Secondary Arterial, Frontage Road, Collector, Local Street, Zonal Connector, Other, Ferry, Walkway
  3) miscellaneous—Ramp, Bikeway, Busway, Light Rail, Heavy Rail.

  There are inconsistent functional classes if there is a mixture of "restricted" links and "surface" links at a node. (This notion can probably be refined further.)

- The network graph is fully connected (i.e., one can reach any node from any other node).
- The network does not contain modal "sources" or "sinks." A modal source (sink) is a node that vehicles of a particular type can leave (enter), but cannot enter (leave).
- The network does not contain unwanted modal "islands." A modal island consists of a set of links for a particular type of vehicle that is disconnected from the rest of the links for that type of vehicle. (There may be some cases, such as for transit routes, where modal islands are desirable.)

Table 3. Link table format.

Column Name	Description	Allowed Values
ID	The ID number of the link.	integer: 1 through 2,147,483,647
NAME	The name of street.	50 characters
NODEA	The ID number of the node at A.	integer: 1 through 2,147,483,647
NODEB	The ID number of the node at B.	integer: 1 through 2,147,483,647
PERMLANESA	The number of lanes on the link heading toward node A, not including pocket lanes.	integer: 0 through 255
PERMLANESB	The number of lanes on the link heading toward node B, not including pocket lanes.	integer: 0 through 255
LEFTPCKTSA	The number of pocket lanes to the left of the permanent lanes heading toward node A.	integer: 0 through 255
LEFTPCKTSB	The number of pocket lanes to the left of the permanent lanes heading toward node B.	integer: 0 through 255

Column Name	Description	Allowed Values
`RGHTPCKTSA`	The number of pocket lanes to the right of the permanent lanes heading toward node A.	integer: 0 through 255
`RGHTPCKTSB`	The number of pocket lanes to the right of the permanent lanes heading toward node B.	integer: 0 through 255
`TWOWAYTURN`	The toggle for a two-way left-turn lane in the center of the link.	one character: `F` = false/no `T` = true/yes
`LENGTH`	The length of the link (in meters).	positive floating-point number
`GRADE`	The percentage grade from node A to node B (uphill being a positive number).	floating-point number between –100 and +100
`SETBACKA`	The setback distance (in meters) from the center of the intersection at node A.	non-negative floating-point number
`SETBACKB`	The setback distance (in meters) from the center of the intersection at node B.	non-negative floating-point number
`CAPACITYA`	The total capacity (in vehicles per hour) for the lanes traveling to node A. This field is obsolete and its value is ignored—it will be deleted in a future TRANSIMS release.	non-negative floating-point number
`CAPACITYB`	The total capacity (in vehicles per hour) for the lanes traveling to node B. This field is obsolete and its value is ignored—it will be deleted in a future TRANSIMS release.	non-negative floating-point number
`SPEEDLMTA`	The default speed limit (in meters per second) for vehicles traveling toward node A.	positive floating-point number
`SPEEDLMTB`	The default speed limit (in meters per second) for vehicles traveling toward node B.	positive floating-point number
`FREESPDA`	The default free-flow speed (in meters per second) for vehicles traveling toward node A.	positive floating-point number
`FREESPDB`	The default free-flow speed (in meters per second) for vehicles traveling toward node B.	positive floating-point number

Column Name	Description	Allowed Values
FUNCTCLASS	The functional class of the link. A link that permits both road and rail traffic should be coded with the roadway class.	ten characters: FREEWAY = freeway XPRESSWAY = expressway PRIARTER = primary arterial SECARTER = secondary arterial FRONTAGE = frontage road COLLECTOR = collector LOCAL = local street RAMP = freeway ramp ZONECONN = zonal connector OTHER = other WALKWAY = walk only BIKEWAY = bicycle only BUSWAY = bus only roadway LIGHTRAIL = light rail only HEAVYRAIL = heavy rail FERRY = ferry
THRUA	The default through link connected at node A. A zero indicates there is no through link.	integer: 0 through 2,147,483,647
THRUB	The default through link connected at node B. A zero indicates there is no through link.	integer: 0 through 2,147,483,647
COLOR	The color number for the link (all of the links connected to a given link must have different colors). This field is obsolete and its value is ignored—it will be deleted in a future TRANSIMS release.	integer: 1 through 63
VEHICLE	The vehicle types (modes) allowed for use this link.	character string separated by slashes: WALK = walking allowed AUTO = private auto TRUCK = motor carrier BICYCLE = bicycle TAXI = paratransit BUS = bus TROLLEY = trolley STREETCAR = streetcar LIGHTRAIL = light rail transit RAPIDRAIL = rail rapid transit REGIONRAIL = regional rail
NOTES	A character string used for data quality annotations; free format (may be blank).	255 characters

Table 4. Functional classes for links.

Class	Interpretation
Freeway	A divided, arterial highway for through traffic with full control of access. Full access control means the authority to control access is exercised to • give preference to through traffic by providing access connections with selected public roads • but prohibiting grade crossings and/or direct private driveway connections.
Expressway	A divided, arterial highway for through traffic with partial control of access. Partial control of access means that some authority is exercised to control access in the manner described above, but there are crossings at grade or direct private driveway connections.
Primary Arterial	A major arterial roadway with intersections at grade crossings and direct access to abutting property and on which geometric design and traffic-control measures are used to expedite safe movement of through traffic.
Secondary Arterial	A minor arterial roadway with intersections at grade crossings and direct access to abutting property and on which geometric design and traffic-control measures are used to expedite safe movement of through traffic.
Frontage Road	An arterial that runs parallel to a freeway or expressway.
Collector Street	A roadway on which vehicular traffic is given preferential right of way. These streets have entrances to which vehicular traffic from intersecting roadways is required by law to yield right-of-way to vehicles as a result of either a stop sign or a yield sign (when such signs are erected).
Local Street	A street or road primarily used to access residence, business, or other abutting property.
Freeway Ramp	A unidirectional roadway that connects a freeway or expressway to an arterial.
Zonal Connector	An imaginary (non-physical) connection to or from the centroid of a traffic analysis zone.
Other	Any roadway not fitting the above definitions.
Walkway	A street restricted to use by pedestrians.
Busway	A street restricted to use by buses.
Light Rail	A roadbed restricted to use by light rail cars.
Heavy Rail	A roadbed restricted to use by heavy rail cars.
Ferry	A waterway crossed by ferry.

1.1.3 Speed Table

Entries in the speed table are required only when the speed limit or free speed for a link varies for different types of vehicles allowed to use the link. The speeds that appear in the Link Tables are used as defaults for any vehicle types not specified in a record in the speed table.

Table 5 specifies the format for the speed table. To validate a speed table, verify the following:

- The field names and types are correct.
- The data values are in the legal ranges.
- The node and link references are correct.
- The vehicle types are consistent with the vehicle types allowed on the link.

Table 5. Speed table format.

Column Name	Description	Allowed Values
`LINK`	The ID number of the link with multiple speeds.	integer: 1 through 2,147,483,647
`NODE`	The ID number of the node toward which lanes are headed.	integer: 1 through 2,147,483,647
`SPEEDLMT`	The speed limit (in meters per second) for vehicles.	positive floating-point number
`FREESPD`	Free-flow speed (in meters per second) for vehicles.	positive floating-point number
`VEHICLE`	The vehicle type to which speeds apply.	character string separated by slashes: `AUTO` = private auto `TRUCK` = motor carrier `BICYCLE` = bicycle `TAXI` = paratransit `BUS` = bus `TROLLEY` = trolley `STREETCAR` = streetcar `LIGHTRAIL` = light-rail transit `RAPIDRAIL` = rail-rapid transit `REGIONRAIL` = regional rail

Column Name	Description	Allowed Values
`STARTTIME`	The starting time for the speeds. This field is ignored in the current TRANSIMS release—current tables should not include records depending on the `STARTTIME`/`ENDTIME` fields.	character string with the day of week: `SUN` = Sunday `MON` = Monday `TUE` = Tuesday `WED` = Wednesday `THU` = Thursday `FRI` = Friday `SAT` = Saturday `WKE` = any weekend day `WKD` = any weekday `ALL` = any day The day is followed by the time of day (on a 24-hour clock). For example, `WKD13:20` is any weekday at 1:20 in the afternoon.
`ENDTIME`	The ending time for the speeds. This field is ignored in the current TRANSIMS release—current tables should not include records depending on the `STARTTIME`/`ENDTIME` fields.	specified like `STARTTIME`
`NOTES`	A character string used for data quality annotations; free format (may be blank).	255 characters

1.1.4 Pocket Lane Table

Table 6 specifies the format for the pocket lane table. To validate a pocket lane table, verify the following:

- The field names and types are correct.
- The data values are in the legal ranges.
- The IDs are unique.
- The node and link references are correct.
- The lane number is that of a valid pocket lane.
- The offset and length are consistent with the setbacks and length of the link.
- None of the pockets overlap.
- All of the pocket lanes specified in the link table are present.

Table 6. Pocket lane table format.

Column Name	Description	Allowed Values
ID	The ID number of the pocket lane.	integer: 1 through 2,147,483,647
NODE	The ID number of the node toward which the pocket lane leads.	integer: 1 through 2,147,483,647
LINK	The ID number of the link on which the pocket lane lies.	integer: 1 through 2,147,483,647
OFFSET	The starting position of the pocket lane, measured (in meters) from NODE (applicable to pullout pockets only).	non-negative floating-point number
LANE	The lane number of the pocket lane.	integer: 1 through 255
STYLE	The type of the pocket lane.	one character: T = turn pocket P = pull-out pocket M = merge pocket
LENGTH	The length of the pocket lane (in meters). Turn and merge pockets always start or end at the appropriate limit line.	positive floating-point number
NOTES	A character string used for data quality annotations; free format (may be blank).	255 characters

1.1.5 Lane-use Table

Entries in the lane-use table are required only when a lane has restrictions for certain vehicle types. The vehicle types specified in the Link Table are permitted unrestricted use of all lanes on the link when there is no record in the lane-use table.

Table 7 specifies the format for the lane-use table. To validate a lane-use table, verify the following:

- The field names and types are correct.
- The data values are in the legal ranges.
- The node, link, and lane references are correct.
- The vehicle types to which restrictions apply are consistent with the vehicle types allowed in the link.

The `VEHICLE` field may designate `HOV` restrictions in addition to the usual vehicle types. For example, to indicate that only buses and autos with at least two occupants are allowed to use a particular lane, specify `BUS/AUTO/HOV2` for the `VEHICLE` field This information will override the vehicle information obtained from the Link Table for the particular lane only.

Table 7. Lane-use table format.

Column Name	Description	Allowed Values
`NODE`	The ID number of the node toward which the lane leads.	integer: 1 through 2,147,483,647
`LINK`	The ID number of the link on which the lane lies.	integer: 1 through 2,147,483,647
`LANE`	The lane number.	integer: 1 through 255
`VEHICLE`	The vehicle type to which restriction applies.	character string separated by slashes: `HOV2` = high-occupancy vehicle with two or more occupants `HOV3` = high-occupancy vehicle with three or more occupants `HOV4` = high-occupancy vehicle with four or more occupants `BICYCLE` = bicycle `AUTO` = private auto `TRUCK` = motor carrier `BUS` = bus `TROLLEY` = trolley `STREETCAR` = streetcar `LIGHTRAIL` = light rail transit `RAPIDRAIL` = rail rapid transit `REGIONRAIL` = regional rail
`RESTRICT`	The type of lane restriction.	one character: `O` = only this vehicle type may use lane `R` = lane required to be used by this vehicle type `N` = lane not allowed to be used by this vehicle type

Column Name	Description	Allowed Values
`STARTTIME`	The starting time for the restriction. This field is ignored in the current TRANSIMS release—current tables should not include records depending on the `STARTTIME/ENDTIME` fields.	character string with the day of week: `SUN` = Sunday `MON` = Monday `TUE` = Tuesday `WED` = Wednesday `THU` = Thursday `FRI` = Friday `SAT` = Saturday `WKE` = any weekend day `WKD` = any weekday `ALL` = any day The day is followed by the time of day (on a 24-hour clock). For example, `WKD13:20` is any weekday at 1:20 in the afternoon
`ENDTIME`	The ending time for the restriction. This field is ignored in the current TRANSIMS release—current tables should not include records depending on the `STARTTIME/ENDTIME` fields.	specified like `STARTTIME`
`NOTES`	A character string used for data quality annotations; free format (may be blank).	255 characters

1.1.6 Parking Table

Table 8 specifies the format for the parking table. To validate a parking table, verify the following:

- The field names and types are correct.
- The data values are in the legal ranges.
- The IDs are unique.
- The node and link references are correct.
- The offset is consistent with the setbacks and length of the link.
- The vehicle types allowed for the parking are consistent with the vehicle types allowed on the link.

Table 8. Parking table format.

Column Name	Description	Allowed Values
`ID`	The ID number of the parking place.	integer: 1 through 2,147,483,647
`NODE`	The ID number of the node toward which vehicles are traveling.	integer: 1 through 2,147,483,647

Column Name	Description	Allowed Values
LINK	The ID number of the link on which the parking place lies.	integer: 1 through 2,147,483,647
OFFSET	The location of the entrance from the link to the parking place, measured (in meters) from NODE.	non-negative floating-point number
STYLE	The type of parking place.	five characters: PRSTR = parallel on street HISTR = head in on street DRVWY = driveway LOT = parking lot BNDRY = network boundary PARKRIDE = park & ride lot
CAPACITY	The number of vehicles the parking place can accommodate; zero for unlimited capacity.	integer: 0 through 65,535
GENERIC	A toggle that indicates whether the parking place represents generic parking (not an actual driveway, lot, etc., but a group/aggregate of them used to simplify modeling).	one character: T = true/yes F = false/no
VEHICLE	The type of vehicle allowed to park at the parking place.	character string separated by slashes: AUTO = private auto TRUCK = motor carrier BICYCLE = bicycle TAXI = paratransit BUS = bus TROLLEY = trolley STREETCAR = streetcar LIGHTRAIL = light-rail transit RAPIDRAIL = rail-rapid transit REGIONRAIL = regional rail ANY = any vehicle type

Column Name	Description	Allowed Values
STARTTIME	The starting time for parking. This field is ignored in the current TRANSIMS release—current tables should not include records depending on the STARTTIME/ENDTIME fields.	character string with the day of week: SUN = Sunday MON = Monday TUE = Tuesday WED = Wednesday THU = Thursday FRI = Friday SAT = Saturday WKE = any weekend day WKD = any weekday ALL = any day The day is followed by the time of day (on a 24-hour clock). For example WKD13:20 is any weekday at 1:20 in the afternoon
ENDTIME	The ending time for parking. This field is ignored in the current TRANSIMS release—current tables should not include records depending on the STARTTIME/ENDTIME fields.	specified like STARTTIME
NOTES	A character string used for data quality annotations; free format (may be blank).	255 characters

1.1.7 Barrier Table

☛ *This feature is not implemented in this version.*

Table 9 specifies the format for the barrier table. The contents of this table are ignored by the current TRANSIMS release. To validate a barrier table, verify the following:

- The field names and types are correct.
- The data values are in the legal ranges.
- The IDs are unique.
- The node, link, and lane references are correct.
- The offset and length are consistent with the setbacks and length of the link.

Table 9. Barrier table format.

Column Name	Description	Allowed Values
ID	The ID number of the barrier.	integer: 1 through 2,147,483,647
NODE	The ID number of the node toward which vehicles are traveling.	integer: 1 through 2,147,483,647

Column Name	Description	Allowed Values
`LINK`	The ID number of the link on which the barrier lies.	integer: 1 through 2,147,483,647
`OFFSET`	The starting position of the barrier, which is measured (in meters) from `NODE`.	non-negative floating-point number
`LANE`	The lane number of lane to the left of the barrier.	integer: 0 through 255
`STYLE`	The type of barrier.	ten characters: `CURB` = curb `BARRIER` = barrier `GRADESEP` = grade separation `STRIPE` = painted stripe `TEMPORARY` = temporary barrier
`LENGTH`	The length of the barrier (in meters).	positive floating-point number
`NOTES`	A character string used for data quality annotations; free format (may be blank).	255 characters

1.1.8 Transit Stop Table

Table 10 specifies the format for the transit stop table. To validate a transit stop table, verify the following:

- The field names and types are correct.
- The data values are in the legal ranges.
- The IDs are unique.
- The node and link references are correct.
- The offset is consistent with the setbacks and length of the link.
- The vehicle types allowed for the transit stop are consistent with the vehicle types allowed on the link.

Table 10. Transit stop table format.

Column Name	Description	Allowed Values
`ID`	The ID number of the stop.	integer: 1 through 2,147,483,647
`NAME`	The name of the stop.	50 characters
`NODE`	The ID number of the node toward which vehicles are traveling.	integer: 1 through 2,147,483,647
`LINK`	The ID number of the link on which the stop takes place.	integer: 1 through 2,147,483,647
`OFFSET`	The location of the stop, which is measured (in meters) from `NODE`.	non-negative floating-point number

Column Name	Description	Allowed Values
VEHICLE	The types of vehicles for which this is a stop.	character string separated by slashes: BUS = bus TROLLEY = trolley STREETCAR = streetcar LIGHTRAIL = light-rail transit RAPIDRAIL = rail-rapid transit REGIONRAIL = regional rail
STYLE	The type of stop.	ten characters: STOP = stop (no station) STATION = station
CAPACITY	The number of vehicles the stop can simultaneously handle; zero for unlimited capacity.	integer: 0 through 65,535
NOTES	A character string used for data quality annotations; free format (may be blank).	255 characters

1.1.9 Lane Connectivity Table

Table 11 specifies the format for the lane connectivity table. To validate a lane connectivity table, verify the following:

- The field names and types are correct.
- The data values are in the legal ranges.
- The node, link, and lane references are correct.
- Each lane has at least one incoming and at least one outgoing connection.

Table 11. Lane connectivity table format.

Column Name	Description	Allowed Values
NODE	The ID number of the node.	integer: 1 through 2,147,483,647
INLINK	The ID number of the incoming link.	integer: 1 through 2,147,483,647
INLANE	The lane number of the incoming lane.	integer: 1 through 255
OUTLINK	The ID number of the outgoing link.	integer: 1 through 2,147,483,647
OUTLANE	The lane number of the outgoing lane.	integer: 1 through 255
NOTES	A character string used for data quality annotations; free format (may be blank).	255 characters

1.1.10 Turn Prohibition Table

☞ *This feature is not implemented in this version.*

Entries in the turn prohibition table are required when particular movements at a node are prohibited only at certain times of the day. The lane connectivity table specifies the allowed and prohibited movements that are otherwise in effect at a node. The contents of this table are ignored by the current TRANSIMS release because time-of-day restrictions are not currently supported.

Table 12 specifies the format for the turn prohibition table. To validate a turn prohibition table, verify the following:

- The field names and types are correct.
- The data values are in the legal ranges.
- The node and link references are correct.

Table 12. Turn prohibition table format.

Column Name	Description	Allowed Values
NODE	The ID number of the node.	integer: 1 through 2,147,483,647
INLINK	The ID number of the incoming link.	integer: 1 through 2,147,483,647
OUTLINK	The ID number of the outgoing link.	integer: 1 through 2,147,483,647
STARTTIME	The starting time for the prohibition. This field is ignored in the current TRANSIMS release—current tables should not include records depending on the STARTTIME/ENDTIME fields.	character string with the day of week: SUN = Sunday MON = Monday TUE = Tuesday WED = Wednesday THU = Thursday FRI = Friday SAT = Saturday WKE = any weekend day WKD = any weekday ALL = any day This day is followed by the time of day (on a 24-hour clock). For example WKD13:20 is any weekday at 1:20 in the afternoon
ENDTIME	The ending time for the prohibition. This field is ignored in the current TRANSIMS release—current tables should not include records depending on the STARTTIME/ENDTIME fields.	specified like STARTTIME
NOTES	A character string used for data quality annotations; free format (may be blank).	255 characters

1.1.11 Unsignalized Node Table

Table 13 specifies the format for the unsignalized node table. To validate an unsignalized node table, verify the following:

- The field names and types are correct.
- The data values are in the legal ranges.
- The node and link references are correct.
- Each incoming link entering an unsignalized node has a record.

Table 13. Unsignalized node table format.

Column Name	Description	Allowed Values
NODE	The ID number of the node.	integer: 1 through 2,147,483,647
INLINK	The ID number of the incoming link.	integer: 1 through 2,147,483,647
SIGN	The type of sign control on the link.	one character: S = stop Y = yield N = none
NOTES	A character string used for data-quality annotations; free format (may be blank).	255 characters

1.1.12 Signalized Node Table

Table 14 specifies the format for the signalized node table. To validate a signalized node table, verify the following:

- The field names and types are correct.
- The data values are in the legal ranges.
- The node references are correct.
- The plan references are correct.
- Each node has either one signalized or one unsignalized control.
- All plans are used.
- The start times are valid.

Table 14. Signalized node table format.

Column Name	Description	Allowed Values
NODE	The ID number of the node.	integer: 1 through 2,147,483,647
TYPE	The type of the signal.	one character: T = timed A = actuated
PLAN	The ID number of a timing plan.	integer: 1 through 65,535
OFFSET	Relative offset (in seconds) for coordinated signals.	non-negative floating-point number
STARTTIME	The starting time for the plan. This field is ignored in the current TRANSIMS release—current tables should not include records depending on the STARTTIME/ENDTIME fields.	a character string with the day of week: SUN = Sunday MON = Monday TUE = Tuesday WED = Wednesday THU = Thursday FRI = Friday SAT = Saturday WKE = any weekend day WKD = any weekday ALL = any day The day is followed by the time of day (on a 24-hour clock). For example WKD13:20 is any weekday at 1:20 in the afternoon
COORDINATR	The ID number of coordinator for the signal; equivalent to NODE number if signal is isolated.	integer: 1 through 2,147,483,647
RING	A single or dual ring, required only for TYPE = A.	one character: S = single D = dual
ALGORITHM	The control algorithm used by signal, required only for TYPE = A.	ten characters
NOTES	A character string used for data quality annotations; free format (may be blank).	255 characters

1.1.13 Phasing Plan Table

Table 15 specifies the format for the phasing plan table. To validate a phasing plan table, verify the following:

- The field names and types are correct.
- The data values are in the legal ranges.
- The plan, phase, node, and link references are correct.
- Each incoming and outgoing link is controlled.

Table 15. Phasing plan table format.

Column Name	Description	Allowed Values
`NODE`	The ID number of the node.	integer: 1 through 2,147,483,647
`PLAN`	The ID number of the timing plan.	integer: 1 through 65,535
`PHASE`	The phase number.	integer: 1 through 255
`INLINK`	The ID number of the incoming link.	integer: 1 through 2,147,483,647
`OUTLINK`	The ID number of the outgoing link.	integer: 1 through 2,147,483,647
`PROTECTION`	The movement protection indicator.	one character: `P` = protected `U` = unprotected `S` = unprotected after stop
`DETECTORS`	The ID number of detectors related to this movement. This is required only for actuated controls.	string of detector IDs, separated by slashes
`NOTES`	A character string used for data quality annotations; free format (may be blank).	255 characters

1.1.14 Timing Plan Table

Table 16 specifies the format for the timing plan table. To validate a timing plan table, verify the following:

- The field names and types are correct.
- The data values are in the legal ranges.
- The (plan, phase) pairs are unique.
- The time values are consistent.
- The phase sequence references existent phases.

Table 16. Timing plan table format.

Column Name	Description	Allowed Values
PLAN	The ID number of timing plan.	integer: 1 through 65,535
PHASE	The phase number.	integer: 1 through 255
NEXTPHASES	The phase number(s) of the next phase(s) in sequence.	string of phase numbers, separated by slashes
GREENMIN	The minimum length (in seconds) of the green interval, or fixed green length for timed signal.	non-negative floating-point number
GREENMAX	The maximum length (in seconds) of the green interval.	non-negative floating-point number
GREENEXT	The length (in seconds) of the green extension interval.	non-negative floating-point number
YELLOW	The length (in seconds) of the yellow interval.	non-negative floating-point number
REDCLEAR	The length (in seconds) of the red clearance interval.	non-negative floating-point number
GROUPFIRST	For pre-timed or single ring: 1 if first phase, 0 if not first phase; for dual ring: number of phase group for which this phase is first phase, 0 if not first phase in the phase group.	integer: 0 through 255
NOTES	A character string used for data quality annotations; free format (may be blank).	255 characters

1.1.15 Detector Table

Table 17 specifies the format for the detector table. To validate a detector table, verify the following:

- The field names and types are correct.
- The data values are in the legal ranges.
- The IDs are unique.
- The node, link, and lane references are correct.
- The offset and length are consistent with the setbacks and length of the link.

Table 17. Detector table format.

Column Name	Description	Allowed Values
`ID`	The ID number of the detector.	integer: 1 through 2,147,483,647
`NODE`	The ID number of the node toward which vehicles are traveling.	integer: 1 through 2,147,483,647
`LINK`	The ID number of the link on which the detector lies.	integer: 1 through 2,147,483,647
`OFFSET`	The starting position of the detector, which is measured (in meters) from `NODE`.	non-negative floating-point number
`LANEBEGIN`	The lane number of lane at which the detector begins.	integer: 1 through 255
`LANEEND`	The lane number at which the detector ends, equal to `LANEBEGIN` for detector that lies on single lane.	integer: 1 through 255
`LENGTH`	The length of the detector (in meters).	non-negative floating-point number
`STYLE`	The type of detector.	ten characters: `PRESENCE` = sense vehicles on detector `PASSAGE` = sense vehicles crossing detector
`COORDINATR`	ID number of coordinators interested in detector output.	string of coordinator IDs separated by slashes
`CATEGORY`	The parameters for the defects this type of detector exhibits.	ten characters: must match last characters of `NET_DETECTOR_*` configuration file keys. A value of 0 may be used to specify no defects.
`NOTES`	A character string used for data quality annotations; free format (may be blank).	255 characters

1.1.16 Signal Coordinator Table

☞ *This feature is not implemented in this version.*

Table 18 specifies the format for the signal coordinator table. To validate a signal coordinator table, verify the following:

- The field names and types are correct.
- The data values are in the legal ranges.
- The IDs are unique.

Table 18. Signal coordinator table format.

Column Name	Description	Allowed Values
`ID`	The ID number of the signal coordinator.	integer: 1 through 2,147,483,647
`TYPE`	The type of coordinator.	ten characters: values to be determined
`ALGORITHM`	The control algorithm used by coordinator. This field is ignored in the current TRANSIMS release.	ten characters: values to be determined
`NOTES`	A character string used for data quality annotations; free format (may be blank).	255 characters

1.1.17 Activity Locations Table

Table 19 specifies the format for the activity location table. To validate an activity location table, verify the following.

- The field names and types are correct.
- The data values are in the legal ranges.
- The IDs are unique.
- The node and link references are correct.
- The offset is consistent with the setbacks and lengths of the links.
- The layer is consistent with the vehicle types allowed on the link.
- The names of any optional user-defined fields are unique within the table.

Table 19. Activity locations table format.

Column Name	Description	Allowed Values
`ID`	The ID number of the activity location.	integer: 1 through 2,147,483,647
`NODE`	The ID number of the node toward which vehicles are traveling (the location is taken to be on the right side of the street when headed this direction).	integer: 1 through 2,147,483,647
`LINK`	The ID number of the link on which the activity location lies.	integer: 1 through 2,147,483,647
`OFFSET`	The location of the entrance from the link to the activity location, which is measured (in meters) from `NODE`.	non-negative floating-point number

`LAYER`	The modal "layer" on which the activity location resides.	character string:
		`AUTO`
		`BUS`
		`LIGHTRAIL`
		`WALK`

`EASTING`	The x-coordinate of the activity location (in meters, UTM grid system).	floating-point number
`NORTHING`	The y-coordinate of the activity location (in meters, UTM grid system).	floating-point number
`ELEVATION`	The z-coordinate of the activity location (in meters, UTM grid system).	floating-point number
`optional field 1`	The first optional field related to land use.	floating-point number
`optional field 2`	The second optional field related to land use.	floating-point number
`optional field n`	The n^{th} optional field related to land use.	floating-point number
`NOTES`	A character string used for data quality annotations; free format (may be blank).	255 characters

A maximum of 25 user-defined fields optionally may be included in the table between the `ELEVATION` and `NOTES` fields. These optional fields are typically related to land use, but could be anything the user wishes to specify about an activity location. The column names may be up to 32 characters in length. The presence of any optional fields is detected by the `NetReadActivityLocationHeader()` function. This implies that the header for the activity location table must be read by this function rather than by `NetReadHeader()` or `NetSkipHeader()`, whether or not optional fields are included.

1.1.18 Process Links Table

Table 20 specifies the format for a process links table. To validate a process link table, verify the following:

- The field names and types are correct.
- The data values are in the legal ranges.
- The IDs are unique.
- The "from" and "to" accessory references are correct.

Table 20. Process links table format.

Column Name	Description	Allowed Values
`ID`	The ID number of the virtual link.	integer: 1 through 2,147,483,647
`FROMID`	The ID number of the accessory from which the virtual link leaves.	integer: 1 through 2,147,483,647
`FROMTYPE`	The type of accessory from which the virtual link leaves.	character string: `ACTIVITY` `PARKING` `TRANSIT`
`TOID`	The ID number of the accessory to which the virtual link leads.	integer: 1 through 2,147,483,647
`TOTYPE`	The type of accessory to which the virtual link leads.	character string: `ACTIVITY` `PARKING` `TRANSIT`
`DELAY`	The time delay (measured in seconds) incurred when traveling across the virtual link.	non-negative floating-point number
`COST`	The cost (measured in arbitrary units) incurred when traveling across the virtual link. Note that although the costs are measured for arbitrary units, the units must be the same for the whole data table.	non-negative floating-point number
`NOTES`	A character string used for data quality annotations; free format (may be blank).	255 characters

1.2 Utility Programs

Several utility programs related to network data files are available.

1.2.1 *ReadNetwork* Utility

The *ReadNetwork* utility reads a specified set of network tables into memory and constructs C++ network objects out of it. It verifies that a network can be read by the Route Planner and the Traffic Microsimulator without actually running those programs. If a logfile is not specified, output is written to standard output.

1.2.1.1 Usage

```
ReadNetwork <configuration file> [<logfile>]
```

1.3 Files

Table 21 lists the network library files.

Table 21. Network library files.

Type	File Name	Description
Binary Files	*libTIO.a*	The TRANSIMS Interfaces library.
Utilities	*ReadNetwork*	The network data file reader.
	ValidateNetwork	The network data file validator.
	SetupNetwork	The tool for creating empty and test network data files.
	CleanupNetwork	The tool for removing empty and test network data files.
Source Files	*netio.c*	The network data structures and interface functions file.
	netio.h	The network interface functions source file.

1.4 Configuration File Keys

Table 22 and Table 23 list the TRANSIMS configuration file keys that specify the location of network data files and the definitions of network-related parameters. The detector defect keys (Table 23) provide parametric definitions for categories of detectors. The category of each detector is specified in the `CATEGORY` field in the Detector Table (see Table 17). In the detector defect keys, the trailing "c" must be replaced by a string that matches a value in the `CATEGORY` field. All values used for `CATEGORY` (except the special value 0) should be defined with detector defect keys. The default for defect keys that are not defined is to treat the detector as though it is non-defective for the undefined characteristic.

Table 22. Network file configuration file keys.

Configuration File Key	Description
`NET_ACTIVITY_LOCATION_TABLE`	The activity location table name.
`NET_ACTUATED_ALGORITHM_B_BETA`	The velocity factor for actuated algorithm B. Default = 1.0 meters/sec
`NET_ACTUATED_ALGORITHM_B_DENSITY_CONST`	The density factor for actuated algorithm B. Default = 0.0/meter
`NET_ACTUATED_ALGORITHM_B_FLOW_CONST`	The flow factor for actuated algorithm B. Default = 0.1/sec
`NET_BARRIER_TABLE`	The barrier table name.
`NET_DETECTOR_PRESENCE_SAMPLE_TIME`	The presence detector sampling frequency. Default = 1 sec

Configuration File Key	Description
NET_DETECTOR_RETENTION_TIME	The retention time for detections. Detections are retained until all interested signals have examined them once or for NET_DETECTOR_RETENTION_TIME, whichever is longer. Default = 0 sec (i.e., cleared after used once)
NET_DETECTOR_TABLE	The detector table name.
NET_DIRECTORY	The directory where the network files reside.
NET_LANE_CONNECTIVITY_TABLE	The lane connectivity table name.
NET_LANE_USE_TABLE	The lane use table name.
NET_LANE_WIDTH	The default lane width (meters).
NET_LINK_MEDIAN_HALFWIDTH	The default half-width (meters) of the median between lanes on a link. To correspond with the current release of the Output Visualizer, this parameter must be assigned a value of 0.5 * NET_LANE_WIDTH.
NET_LINK_TABLE	The link table name.
NET_NODE_TABLE	The node table name.
NET_PARKING_TABLE	The parking table name.
NET_PHASING_PLAN_TABLE	The phasing plan table name.
NET_POCKET_LANE_TABLE	The pocket lane table name.
NET_PROCESS_LINK_TABLE	The process link table name.
NET_SIGNAL_COORDINATOR_TABLE	The signal coordinator table name.
NET_SIGNALIZED_NODE_TABLE	The signalized node table name.
NET_SPEED_TABLE	The speed table name.
NET_TIMING_PLAN_TABLE	The timing plan table name.
NET_TRANSIT_STOP_TABLE	The transit stop table name.
NET_TURN_PROHIBITION_TABLE	The turn prohibition table name.
NET_UNSIGNALIZED_NODE_TABLE	The unsignalized node table name.

Table 23. Detector defect keys.

Configuration File Key	Description
NET_DETECTOR_ACCELERATION_NOISE_c	The standard deviation of random error in detection acceleration (meters/second/second).
NET_DETECTOR_ACCELERATION_OFFSET_c	The systematic error in detection acceleration (meters/second/second).
NET_DETECTOR_FAILURE_TIME_MEAN_c	The mean time (seconds) between detector catastrophic failures. A value of 0 indicates no failures.
NET_DETECTOR_FALSE_ALARM_PROBABILITY_c	The probability of counting the same detection twice.

Configuration File Key	Description
NET_DETECTOR_FALSE_ALARM_TIME_MEAN_c	The mean time (seconds) between spontaneous false alarms (i.e., recording a detection when no vehicle was there). A value of 0 indicates no spontaneous false alarms.
NET_DETECTOR_INITIAL_FAILURE_PROBABILITY_c	The probability detector is broken at beginning of simulation.
NET_DETECTOR_MISS_ACCELERATION_PROBABILITY_C	The probability of missing the acceleration component of a detection.
NET_DETECTOR_MISS_POSITION_PROBABILITY_c	Probability of missing the position component of a detection.
NET_DETECTOR_MISS_PROBABILITY_c	Probability of detector missing a detection.
NET_DETECTOR_MISS_VELOCITY_PROBABILITY_c	Probability of missing the velocity component of a detection.
NET_DETECTOR_POSITION_NOISE_c	Standard deviation of random error in detection position (meters).
NET_DETECTOR_POSITION_OFFSET_c	Systematic error in detection position (meters).
NET_DETECTOR_VELOCITY_NOISE_c	Standard deviation of random error in detection velocity (meters/second).
NET_DETECTOR_VELOCITY_OFFSET_c	Systematic error in detection velocity (meters/second).
NET-DETECTOR_REPAIR_TIME_MAX_c	Maximum time (seconds) until failed detector is repaired. A value of 0 indicates detector is immediately repaired. A value of -1 indicates no repair.

1.5 Examples

Appendix A provides a network files example.

1.6 Tools

1.6.1 Network Validator

The network validation tool has been enhanced in several respects:

- all tables are now checked
- new tests have been added and old ones have been corrected
- all parameters are set using configuration keys
- the output is in XML format

1.6.1.1 Usage

The network validator is run from the command line with the configuration file name and the output file name as arguments:

```
ValidateNetwork input.cfg output.xml
```

The table below lists the configuration file keys used by the validator.

Table 24. Configuration file keys used by the validator.

Configuration File Key	Description
NET_ACTIVITY_LOCATION_TABLE	The activity location table name.
NET_BARRIER_TABLE	The barrier table name.
NET_DETECTOR_TABLE	The detector table name.
NET_DIRECTORY	The directory where the network files reside.
NET_LANE_CONNECTIVITY_TABLE	The lane connectivity table name.
NET_LANE_USE_TABLE	The lane use table name.
NET_LINK_TABLE	The link table name.
NET_NODE_TABLE	The node table name.
NET_PARKING_TABLE	The parking table name.
NET_PHASING_PLAN_TABLE	The phasing plan table name.
NET_POCKET_LANE_TABLE	The pocket lane table name.
NET_PROCESS_LINK_TABLE	The process link table name.
NET_SIGNAL_COORDINATOR_TABLE	The signal coordinator table name.
NET_SIGNALIZED_NODE_TABLE	The signalized node table name.
NET_SPEED_TABLE	The speed table name.
NET_TIMING_PLAN_TABLE	The timing plan table name.
NET_TRANSIT_STOP_TABLE	The transit stop table name.
NET_TURN_PROHIBITION_TABLE	The turn prohibition table name.
NET_UNSIGNALIZED_NODE_TABLE	The unsignalized node table name.
NETV_ACTIVITY_LOCATION	Whether to validate the activity location table.
NETV_BARRIER	Whether to validate the barrier table.
NETV_DETECTOR	Whether to validate the detector table.
NETV_DISTDIFF	The minimum discrepancy (in meters) allowed between the length of a link and its Euclidean distance. Default = 7.5 meters
NETV_DISTFACT	The maximum ratio allowed between the length of a link and its Euclidean distance. Default = 1.5
NETV_LANE_CONNECTIVITY	Whether to validate the lane connectivity table.
NETV_LANE_USE	Whether to validate the lane use table.
NETV_LINK	Whether to validate the link table.
NETV_MAXLANES	The maximum number of lanes on a link. Default = 4
NETV_MAXLINK	The maximum number of links allowed at a node. Default = 5
NETV_MINLEN	The minimum link length (in meters) allowed. Default = 30 meters

Configuration File Key	Description
NETV_MINLINK	The minimum number of links allowed at a node. Default = 2
NETV_MINOFF	The minimum offset (in meters) allowed for parking locations and transit stops. Default = 15 meters
NETV_MINSPEED	The minimum speed limit or free speed (in meters per second) allowed. Default = 7.5 meters per second
NETV_NODE	Whether to validate the node table.
NETV_NODEDIST	The closest distance (in meters) allowed between two nodes. Default = 15 meters
NETV_PARKING	Whether to validate the parking table.
NETV_PHASING_PLAN	Whether to validate the phasing plan table.
NETV_POCKET_LANE	Whether to validate the pocket lane table.
NETV_PROCESS_LINK	Whether to validate the process link table.
NETV_SIGNAL_COORDINATOR	Whether to validate the signal coordinator table.
NETV_SIGNALIZED_NODE	Whether to validate the signalized node table.
NETV_SPEED	Whether to validate the speed table.
NETV_TIMING_PLAN	Whether to validate the timing plan table.
NETV_TRANSIT_STOP	Whether to validate the transit stop table.
NETV_TURN_PROHIBITION	Whether to validate the turn prohibition table.
NETV_UNSIGNALIZED_NODE	Whether to validate the unsignalized node table.
NETV_WARNINGS	Whether or not to report warnings or informational messages.

The XML output file conforms to the following document type definition (DTD):

```
<!ELEMENT validation (table*)>

<!ELEMENT table      (problem*)>
<!ATTLIST table      type   CDATA                        #REQUIRED>

<!ELEMENT problem    (record?,message?,id*,datum*)>
<!ATTLIST problem    level  (fatal|error|warning|info)  #REQUIRED
                     type   CDATA                        #REQUIRED>

<!ELEMENT record     (#PCDATA)>

<!ELEMENT message    (#PCDATA)>

<!ELEMENT id         (#PCDATA)>
<!ATTLIST id         type   CDATA                        #REQUIRED>

<!ELEMENT datum      (#PCDATA)>
<!ATTLIST datum      type   CDATA                        #REQUIRED>
```

Each `problem` element provides information on a specific problem that was found in a network data table—the next section lists the tests in detail. The `level` attribute distinguishes the severity of the problem: namely,

- `fatal` – The problem prevents the validation from proceeding further (either on the record, table, or network).
- `error` – A simulation using this data will probably fail because of this problem.
- `warning` – Although the simulation will probably run, this problem will cause undesirable results.
- `info` – The data are suspicious, but the problem will not affect the simulation.

The `type` attribute and the sub-elements provide diagnostic details on the problem encountered. The problems for each network table are grouped under `table` elements.

The following fragment of an output file illustrates the organization of the XML:

```
<?xml version='1.0' standalone='yes'?>
<!DOCTYPE validation SYSTEM 'validation.dtd'>
<validation>
  <table type='node'>
    <!-- NETV_NODEDIST = 15 -->
    <problem level='info' type='nodes too close'>
      <record>242</record>
      <id type='node'>252</id>
      <id type='node'>250</id>
      <datum type='distance'>8.87764</datum>
    </problem>
    <problem level='info' type='nodes too close'>
      <record>270</record>
      <id type='node'>280</id>
      <id type='node'>279</id>
      <datum type='distance'>9.21954</datum>
    </problem>
        .
        .
        .
  </table>
  <table type='link'>
    <!-- NETV_MINLEN = 30 -->
    <!-- NETV_DISTDIFF = 7.5 -->
    <!-- NETV_DISTFACT = 1.5 -->
    <!-- NETV_MINSPEED = 7.5 -->
    <!-- NETV_MINLINK = 2 -->
    <!-- NETV_MAXLINK = 5 -->
    <problem level='info' type='inconsistent functional classes'>
      <record>12</record>
      <id type='link'>200003</id>
      <id type='node'>9204</id>
    </problem>
    <problem level='error' type='link too short'>
      <record>76</record>
      <id type='link'>219357</id>
```

```
      <datum type='adjusted length'>17.7</datum>
    </problem>
    <problem level='warning' type='no lanes'>
      <record>123154</record>
      <id type='parking'>123430</id>
      <id type='link'>900131</id>
      <id type='node'>32319</id>
    </problem>
        .
        .
        .
  </table>
  <table type='transit stop'>
  </table>
</validation>
```

Several XSLT-based filters are being provided for summarizing, analyzing, and extracting data from the XML output file:

- `report` – Writes a report similar to the one output by the previous version of the network validator.
- `summarize` – Lists the number of occurrences of each type of error in each table.
- `extract` – Lists the occurrences of a specified type of error in a given table.
- `locate` – Lists the errors associated with a specific node, link, parking, etc. ID.

It is not difficult to write additional filters for specific purposes.

1.6.1.2 Tests

Most of the validation tests performed are specific to the network table being tested. The two tests listed in Table 25 are applied to all tables, however:

Table 25. General tests.

Type of Problem	Description	Level	Data
Illegal value	The specified field has an illegal value.	error	record number detailed message
Invalid format	The format of the table is not correct.	fatal	record number detailed message

The "Type of Problem" column above corresponds to the `type` attribute in the problem element of the XML output; the "Level" column corresponds to the `level` attribute. The following charts list the tests performed for the various network tables. Tests displayed below in a gray font have not yet been implemented. Because the tables are validated in the order given below, severe errors in a table may prevent the validation of subsequent tables.

1.6.1.2.1 Node Table Tests

Table 26. Node table tests.

Type of Problem	Description	Level	Data
Duplicate ID	Another node has the same ID.	fatal	record number
			node ID
Nodes too close	Two nodes are closer than `NETV_NODEDIST` meters.	info	first node ID
			second node ID
			distance

1.6.1.2.2 Link Table Tests

Table 27. Link table tests.

Type of Problem	Description	Level	Data
Duplicate ID	Another link has the same ID.	fatal	record number
			link ID
Node does not exist	The link connects to a node that does not exist in the node table.	fatal	record number
			link ID
			node ID
Link starts and ends on same node	Both ends of the link connect to the same node.	error	record number
			link ID
			node ID
Duplicate connectivity	Another link already connects the same two nodes.	warning	record number
			link ID
			node ID
			node ID
Left pockets without permanent lanes	There are left pockets on the link in a direction where there are no permanent lanes.	error	record number
			link ID
			"toward" node ID
Right pockets without permanent lanes	There are right pockets on the link in a direction where there are no permanent lanes.	error	record number
			link ID
			"toward" node ID
No permanent lanes	There are no lanes in either direction, but vehicles are not allowed on the link.	warning	record number
			link ID
No permanent lanes for vehicles	There are no lanes in either direction, and vehicles are allowed on the link.	error	record number
			link ID
Too many lanes	The link has more than `NETV_MAXLANES` lanes in the specified direction.	info	record number
			link ID
			"toward" node ID
			number of lanes
Length inconsistent with setbacks	The sum of the link's setback distances is greater than the link's length.	error	record number
			link ID
			length
			sum of setback distances
Link too short	The link is less that `NETV_MINLEN` meters long.	error	record number
			link ID

Type of Problem	Description	Level	Data
			length of the link minus the setback distances
Length much shorter than Euclidean distance	The length of the link is more than `NETV_DISTDIFF` meters shorter than the distance between its endpoints.	info	record number link ID length Euclidean distance
Length shorter than Euclidean distance	The length of the link is smaller than the distance between its endpoints.	info	record number link ID length Euclidean distance
Length much longer than Euclidean distance	The length of the link is more than a factor of `NETV_DISTFACT` times longer than the distance between its endpoints.	info	record number link ID length Euclidean distance
No capacity	The capacity of the link is zero in a direction with lanes.	warning	record number link ID "toward" node ID
Speed limit too small	The speed limit is less than `NETV_MINSPEED` meters per second in a direction with lanes.	warning	record number link ID "toward" node ID speed limit
Free speed too small	The free speed is less than `NETV_MINSPEED` meters per second in a direction with lanes.	warning	record number link ID "toward" node ID free speed
Free speed less than speed limit	The free speed is less than the speed limit in a direction with lanes.	warning	record number link ID "toward" node ID free speed speed limit
Inconsistent functional classes	The functional class of the link is not consistent with the functional classes of other links connected to the same node.	info	record number link ID node ID
No travel allowed	No vehicle types are allowed on the link.	warning	record number link ID
Functional class inconsistent with vehicles	The functional class of the link is not consistent with the types of vehicles allowed on the link.	info	record number link ID
No through link	There is no "through" link at the specified node.	warning	record number link ID node ID
Through link does not exist	The "through" link does not exist in the link table.	error	record number link ID node ID through link ID
Through link does not connect	The "through" link does not connect to the specified node at the link's end.	error	record number link ID node ID through link ID
No links for node	No links are connected to the node.	error	node ID
Too few incoming links		warning	node ID

Type of Problem	Description	Level	Data
for node	Less than `NETV_MINLINK` links lead into the node.		number of links
Too few outgoing links for node	Less than `NETV_MINLINK` links lead out from the node.	warning	node ID number of links
Too many incoming links for node	More than `NETV_MAXLINK` links lead into the node. Note that the count of links does not include "duplicate" links that connect the same two nodes.	warning	node ID number of links
Too many outgoing links for node	More than `NETV_MAXLINK` links lead out from the node. Note that the count of links does not include "duplicate" links that connect the same two nodes.	warning	node ID number of links
Node cannot be reached from links	The node is part of an "island" of nodes that are unreachable by the rest of the network.	warning	node ID
Node cannot reach links	The node is part of an "island" of nodes that cannot reach the rest of the network.	warning	node ID
Incoming links for mode without outgoing links	Vehicles of a certain type are allowed to enter a node, but not leave it.	warning	node ID vehicle type
Outgoing links for mode without incoming links	Vehicles of a certain type are allowed to leave a node, but not enter it.	warning	node ID vehicle type

1.6.1.2.3 Speed Table Tests

Table 28. Speed table tests.

Type of Problem	Description	Level	Data
Duplicate record	A record for this direction of this link already exists	error	record number link ID "toward" node ID
Link does not exist	There is no link in the specified direction.	fatal	record number link ID "toward" node ID
No lanes	There are no permanent lanes in this direction.	warning	record number link ID "toward" node ID
No travel allowed	No vehicles are allowed on the link in the specified direction.	warning	record number link ID "toward" node ID
Speed limit too small	The speed limit is less than `NETV_MINSPEED` meters per second in a direction with lanes.	warning	record number link ID "toward" node ID speed limit
Free speed too small	The free speed is less than `NETV_MINSPEED` meters per second in a direction with lanes.	warning	record number link ID "toward" node ID free speed
Free speed less than speed limit	The free speed is less than the speed limit in a direction with lanes.	warning	record number link ID "toward" node ID

Type of Problem	Description	Level	Data
			free speed
			speed limit
Inconsistent vehicle types	The vehicle types are not consistent with those allowed on this link.	warning	record number
			link ID

1.6.1.2.4 Pocket Lane Table Tests

Table 29. Pocket lane table tests.

Type of Problem	Description	Level	Data
Duplicate ID	Another pocket lane has the same ID.	fatal	record number
			pocket lane ID
Link does not exist	There is no link in the specified direction.	fatal	record number
			pocket lane ID
			link ID
			“toward” node ID
Lane does not exist	The specified lane does not exist on the link.	fatal	record number
			pocket lane ID
			link ID
			“toward” node ID
			lane number
Two turn pockets on same lane	There are two turn pockets with the same lane number.	error	record number
			pocket lane ID
			lane number
Two merge pockets on same lane	There are two merge pockets with the same lane number.	error	record number
			pocket lane ID
			lane number
Two pullout pockets on same lane	There are two pullout pockets with the same lane number.	info	record number
			pocket lane ID
			lane number
Offset must be zero	The offset for a turn or merge pocket must be zero.	warning	record number
			pocket lane ID
			offset
Invalid offset	The pullout pocket’s offset is not consistent with the length of the link.	error	record number
			pocket lane ID
			offset
			minimum offset
			maximum offset
Invalid length	The length of the pocket lane is inconsistent with that of the link.	error	record number
			pocket lane ID
			length
Overlapping pocket lanes	The pocket lane overlaps with another one.	error	record number
			first pocket lane ID
			second pocket lane ID
			link ID
			“toward” node ID
Missing pocket lane	There should be the specified pocket lane on this link.	error	link ID
			“toward” node ID
			lane number

1.6.1.2.5 Lane Use Table Tests

Table 30. Lane use table tests.

Type of Problem	Description	Level	Data
Duplicate record	A record for this direction of this link already exists	error	record number link ID "toward" node ID lane number
Link does not exist	There is no link in the specified direction.	fatal	record number link ID "toward" node ID
Lane does not exist	The specified lane does not exist.	fatal	record number link ID "toward" node ID lane number
Inconsistent vehicle types	The vehicle types are not consistent with the vehicles allowed on the link.	warning	record number link ID "toward" node ID lane number vehicle type
Inconsistent restrictions	The restriction is inconsistent with another lane use restriction.	warning	record number link ID "toward" node ID lane number

1.6.1.2.6 Parking Table Tests

Table 31. Parking table tests.

Type of Problem	Description	Level	Data
Duplicate ID	Another parking location has the same ID.	fatal	record number parking ID
Link does not exist	There is no link in the specified direction.	fatal	record number parking ID link ID "toward" node ID
No lanes	There are no lanes on the specified side of the link.	info	record number parking ID link ID "toward" node ID
Invalid offset	The offset is inconsistent with the length of the link.	error	record number parking ID offset minimum offset maximum offset
Offset too small	The offset is too small (less than `NETV_MINOFF` meters from the node) for the Microsimulator—the location will be	warning	record number parking ID offset

Type of Problem	Description	Level	Data
	adjusted during the simulation.		minimum offset
No vehicles allowed	No vehicles are allowed in the parking location.	warning	record number parking ID
Overlapping parking	Another parking location has the same position.	warning	record number first parking ID second parking ID link ID “toward” node ID
Inconsistent vehicle types	The vehicle types are not consistent with the vehicles allowed on the link.	warning	record number parking ID link ID
Inconsistent functional class	The presence of parking is not consistent with the functional class of the link.	info	record number parking ID link ID

1.6.1.2.7 Barrier Table Tests

Table 32. Barrier table tests.

Type of Problem	Description	Level	Data
Duplicate ID	Another barrier has the same ID.	fatal	record number barrier ID
Link does not exist	There is no link in the specified direction.	fatal	record number barrier ID link ID “toward” node ID
Lane does not exist	The specified lane does not exist.	fatal	record number barrier ID link ID “toward” node ID lane number
Invalid offset	The offset is inconsistent with the length of the link.	error	record number barrier ID offset minimum offset maximum offset
Invalid length	The length of the barrier is inconsistent with that of the link.	error	record number barrier ID length
Overlapping barriers	Another barrier has the same position.	warning	record number first barrier ID second barrier ID link ID “toward” node ID

1.6.1.2.8 Transit Stop Table Tests

Table 33. Transit stop table tests.

Type of Problem	Description	Level	Data
Duplicate ID	Two transit stops have the same ID.	fatal	record number transit stop ID
Link does not exist	There is no link in the specified direction.	fatal	record number transit stop ID link ID “toward” node ID
No lanes	There are no lanes on the specified side of the link.	warning	record number transit stop ID link ID “toward” node ID
Invalid offset	The offset is inconsistent with the length of the link	error	record number transit stop ID offset minimum offset maximum offset
Offset too small	The offset is too small (less than `NETV_MINOFF` meters from the node) for the Microsimulator—the location will be adjusted during the simulation.	warning	record number transit stop ID offset minimum offset
No transit vehicles allowed	No transit vehicles are allowed in the transit stop.	warning	record number transit stop ID
Inconsistent vehicle types	The vehicle types are not consistent with the vehicles allowed on the link.	warning	record number link ID “toward” node ID lane number vehicle type
Inconsistent functional class	The presence of a transit stop is not consistent with the functional class of the link.	info	record number transit stop ID link ID
Overlapping transit stops	Another transit stop has the same position.	warning	record number first transit stop ID second transit stop ID link ID “toward” node ID

1.6.1.2.9 Lane Connectivity Table Tests

Table 34. Lane connectivity table tests.

Type of Problem	Description	Level	Data
Duplicate record	The same connectivity is specified by another record.	warning	record number
			node ID
			incoming link ID
			incoming lane number
			outgoing link ID
			outgoing lane number
Incoming link does not exist	There is no incoming link at the specified node.	fatal	record number
			link ID
			node ID
Incoming lane does not exist	The specified incoming lane does not exist.	error	record number
			link ID
			node ID
			lane number
Outgoing link does not exist	There is no outgoing link at the specified node.	fatal	record number
			node ID
			link ID
Outgoing lane does not exist	The specified outgoing lane does not exist.	error	record number
			node ID
			link ID
			lane number
No incoming connectivity	There is no connectivity for the specified incoming lane.	warning	node ID
			link ID
			lane number
No outgoing connectivity	There is no connectivity for the specified outgoing lane.	warning	node ID
			link ID
			lane number

1.6.1.2.10 Turn Prohibition Table Tests

Table 35. Turn prohibition table tests.

Type of Problem	Description	Level	Data
Incoming link does not exist	There is no incoming link at the specified node.	fatal	record number
			link ID
			node ID
Outgoing link does not exist	There is no outgoing link at the specified node.	fatal	record number
			node ID
			link ID
No connectivity	The movement is never allowed.	error	record number
			node ID
			incoming link ID
			outgoing link ID

1.6.1.2.11 Timing Plan Table Tests

Table 36. Timing plan table tests.

Type of Problem	Description	Level	Data
Duplicate phase	The same phase has been specified by another record.	error	record number
			plan ID
			phase number
No next phase	No next phase has been specified	warning	record number
			plan ID
			phase number
Invalid next phase	The specified next phase does not exist in the plan.	error	record number
			plan ID
			phase number
			next phase number
Never next phase	The specified phase is never a next phase in the plan.	warning	plan ID
			phase number
Invalid group first		error	record number
			plan ID
			phase number

1.6.1.2.12 Signal Coordinator Table Tests

Table 37. Signal coordinator table tests.

Type of Problem	Description	Level	Data
Duplicate ID	Another signal coordinator has the same ID.	fatal	record number
			coordinator ID

1.6.1.2.13 Detector Table Tests

Table 38. Detector table tests.

Type of Problem	Description	Level	Data
Duplicate ID	Another detector has the same ID.	fatal	record number
			detector ID
Link does not exist	There is no link in the specified direction.	fatal	record number
			detector ID
			link ID
			“toward” node ID
Begin lane does not exist	The specified begin lane does not exist.	error	record number
			detector ID
			link ID
			“toward” node ID
			lane number
End lane does not exist	The specified end lane does not exist.	error	record number
			detector ID
			link ID
			“toward” node ID
			lane number
Inconsistent begin and end lanes	The begin lane number is greater than the end lane number.	error	record number
			detector ID
			link ID
			“toward” node ID
			begin lane number
			end lane number
Invalid offset	The offset is inconsistent with the length of the link.	error	record number
			detector ID
			offset
			minimum offset
			maximum offset
Invalid length	The length of the detector is inconsistent with that of the link.	error	record number
			detector ID
			length
Coordinator does not exist	The specified signal coordinator does not exist.	error	record number
			detector ID
			coordinator ID

1.6.1.2.14 Signalized Node Table Tests

Table 39. Signalized node table tests.

Type of Problem	Description	Level	Data
Duplicate record	A phasing plan has already been specified for this node.	error	record number node ID
Node does not exist	The specified node does not exist.	error	record number node ID
Plan does not exist	The specified timing plan does not exist.	error	record number node ID plan ID
Timed control is coordinated	A signal coordinator exists for a timed control.	warning	record number node ID coordinator ID
Coordinator does not exist	The specified signal coordinator does not exist.	error	record number node ID coordinator ID
Algorithm does not exist	The specified algorithm does not exist.	error	record number node ID algorithm
Coordinator not used	The signal coordinator is not used by any signal.	warning	coordinator ID

1.6.1.2.15 Phasing Plan Table Tests

Table 40. Phasing plan table tests.

Type of Problem	Description	Level	Data
Duplicate record	The same movement and phase has been specified by another record.	error	record number node ID plan ID phase number incoming link ID outgoing link ID
Plan does not exist	The specified plan has not been assigned to this node.	error	record number node ID plan ID
Phase does not exist	The specified phase does not exist in the plan.	error	record number node ID plan ID phase number
Incoming link does not exist	The specified incoming link does not exist at the node.	error	record number node ID link ID
Outgoing link does not exist	The specified outgoing link does not exist at the node.	error	record number node ID link ID
No connectivity	The movement is never allowed.	warning	record number node ID incoming link ID

Type of Problem	Description	Level	Data
			outgoing link ID
Detector does not exist	The specified detector does not exist.	error	record number
			node ID
			detector ID
No detectors	No detectors have been specified for an actuated signal.	warning	record number
			node ID
Plan not used	The plan is not used by any intersections.	info	plan ID
No phase	No movements have been specified for the phase.	warning	node ID
			phase number
No incoming phase	No movements have been specified for an incoming link.	warning	node ID
			link ID
No outgoing phase	No movements have been specified for an outgoing link.	warning	node ID
			link ID
Detector not used	The detector is not used by any signal.	warning	detector ID

1.6.1.2.16 Unsignalized Node Table Tests

Table 41. Unsignalized node table tests.

Type of Problem	Description	Level	Data
Incoming link does not exist	There is no incoming link at the specified node.	error	record number
			node ID
			link ID
Duplicate record	A sign has already been given for the specified incoming link.	error	record number
			node ID
			link ID
Signal present	A signal exists at the node.	error	node ID
No sign	There is no sign at the specified incoming link.	error	node ID
			link ID

1.6.1.2.17 Activity Location Table Tests

Table 42. Activity location table tests.

Type of Problem	Description	Level	Data
Duplicate ID	Two activity locations have the same ID.	fatal	record number activity location ID
Link does not exist	There is no link in the specified direction.	fatal	record number activity location ID link ID "toward" node ID
No lanes	There are no lanes on the specified side of the link.	warning	record number activity location ID link ID "toward" node ID
Invalid offset	The offset is inconsistent with the length of the link	error	record number activity location ID offset minimum offset maximum offset
Inconsistent layer	The activity location's layer is not consistent with the vehicles allowed on the link.	warning	record number activity location ID layer
Overlapping activity locations	Another activity location has the same position.	warning	record number first activity location ID second activity location ID link ID "toward" node ID

1.6.1.2.18 Process Link Table Tests

Table 43. Process link table tests.

Type of Problem	Description	Level	Data
Duplicate ID	Two process links have the same ID.	error	record number process link ID
Duplicate record	A process link between these activity locations has already been specified.	error	record number process link ID "from" accessory ID "to" accessory ID
From accessory does not exist	The specified "from" accessory does not exist.	fatal	record number process link ID "from" accessory ID
To accessory does not exist	The specified "to" accessory does not exist.	fatal	record number process link ID "to" accessory ID
Accessory not used	The specified accessory is not used.	warning	accessory ID

1.6.2 Lane Connectivity Generator

LaneConnectivity is a program that creates a TRANSIMS lane connectivity table as defined in Table 11. This table provides information that indicates which outgoing lanes and links are valid for subsequent movements of vehicles on lanes entering an intersection. The connectivity generation algorithm, described in detail below, uses information from the TRANSIMS node, link, pocket lane, and use tables that describe the geometry of the network, the placement of permanent and pocket lanes, and the functional classes of links. It is strongly recommended to run the *ValidateNetwork* program and correct identified network errors in these tables prior to running *LaneConnectivity*.

LaneConnectivity may be used in the absence of any lane connectivity information to generate connectivity at all nodes. It may also be used in conjunction with existing lane connectivity information and will then generate connectivity for only those nodes that still require connectivity. When used in this mode, existing files and generated files must be manually concatenated before being used as input to the applications.

1.6.2.1 Usage

```
LaneConnectivity <configuration file> <log file>
```

1.6.2.2 Configuration File Keys

Valid values for the following network configuration file keys must be provided in the configuration file.

```
NET_DIRECTORY
NET_LANE_USE_TABLE
NET_LINK_TABLE
NET_NODE_TABLE
NET_POCKET_LANE_TABLE
```

If the following key specifies an existing data file, the information will be used by *LaneConnectivity*. Otherwise, the key should specify an empty data file with a proper header.

```
NET_LANE_CONNECTIVITY_TABLE
```

Additional configuration keys used only by LaneConnectivity are defined in Table 44.

Table 44. Additional configuration keys used only by *LaneConnectivity*.

Configuration File Key	Description
NET_LANE_CONNECTOR_DETAIL	Indicates whether details about the connectivity generation will be printed. Valid values are TRUE or FALSE. Default = FALSE
NET_LANE_CONNECTOR_SINGLE_NODE	Indicates whether connectivity is created for only one node. This may be useful for testing. A valid value is a node ID.
NET_NEW_LANE_CONNECTIVITY_TABLE	The name of the lane connectivity table that will be created. No default value.

1.6.2.3 Log File Output

The log file contains a count of the number of nodes in an existing lane connectivity file if one was used as input, and a count of the number of nodes for which lane connectivity was generated. It may optionally contain details about the generation process.

Note that the detailed output is voluminous and generally only useful for debugging.

1.6.2.4 Traffic Control Table Output

The following table is produced by LaneConnectivity. This file must be manually concatenated with any existing connectivity file that was used as input by *LaneConnectivity*.

1.6.2.4.1 Lane Connectivity Table

The lane connectivity table is written to the file named by the NET_NEW_LANE_CONNECTIVITY_TABLE configuration file key.

1.6.2.5 Lane Connectivity Algorithm

The lane connectivity algorithm has been modified since this description was written. The description in this document gives the general flavor of how the lane connectivities are generated, but it does not quite correspond to the most recent version of the coded algorithm. When the updated documentation is complete, it will be made available.

To create the lane connectivities, we examine intersections and determine each link's relative direction into the intersection, using the coordinates of the link's end points (nodes). Each incoming and outgoing lane of an intersection must have connectivity with at least one outgoing or incoming lane, respectively. To guarantee connectivity for all incoming lanes, the algorithm is written from the perspective of the normal traffic flow—i.e., in the normal traveling direction. To guarantee connectivity for all outgoing lanes, a comparable algorithm is written from the perspective of traffic flow in the reverse direction. To reuse some techniques of the first algorithm in the latter case, the network is inverted to reverse the incoming and outgoing concepts. The connectivities produced by the second algorithm must be reversed again, and then a union of the connectivities from the two algorithms is formed to complete the process.

1.6.2.5.1 Rotating the Intersection

For each link into the intersection, the lane connectivities for the incoming lanes (i.e., the lanes entering the intersection) are calculated by rotating the intersection (or comparably, by rotating the coordinate system) such that the *subject* link's incoming lanes are headed north as shown in Fig. 1. The other links' incoming lane connectivities are not determined until they, in turn, are rotated into the north pointing position and become the *subject* link.

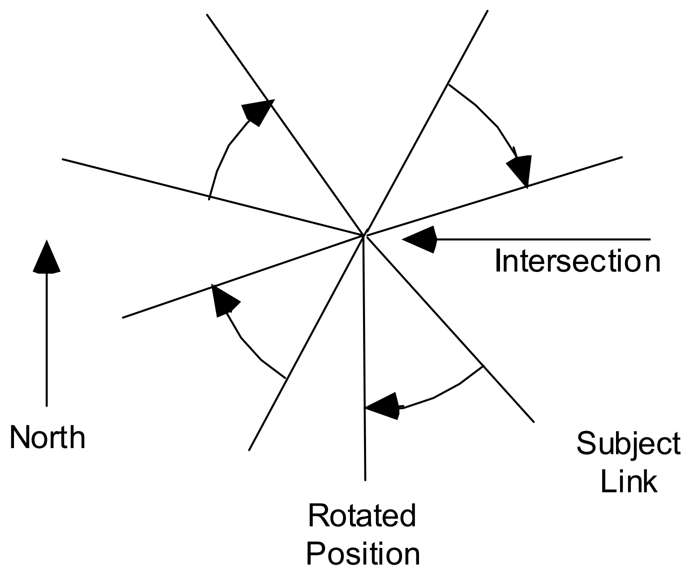

Fig. 1 Intersection rotation with the subject link in the north-pointing position.

Rotating the intersection is simply a transformation of coordinates with a slight modification. Because we rotate the subject link to 180°, the transformation equations differ in sign from the usual set,

$$x' = -x\cos(\Phi_s) + y\sin(\Phi_s),$$
$$y' = -y\cos(\Phi_s) - x\sin(\Phi_s)\ and$$
$$\cos(\Phi') = \frac{y'}{L},$$

where Φ_S is the subject link's angle, x and y are the relative coordinates of the link's node that is not the intersection, and L is the subject link's length. One can compute the trigonometric functions for the subject link because we have the link's endpoint coordinates.

1.6.2.5.2 Lane Connectivity Algorithm

Considerable analysis of the intersection characteristics is necessary before the lane connectivities are created. The algorithm to determine the lane connectivities for the subject link proceeds in five steps:

1) Find the most likely through link.
2) Determine through lanes, left lanes, and right lanes.
3) Create through lane connectivities.
4) Create left lane connectivities.
5) Create right lane connectivities.

1.6.2.5.3 Finding the Most Likely Through Link

The search for the most likely through link (Fig. 2) proceeds by examining each link's cosine, with a few caveats, and selecting the one with the maximum cosine. First, only those links with outgoing lanes and angles less than 90° (cosines greater than zero) are located. Next, since the most likely through link probably will have the same functional class as the subject link, our search initially is restricted to links with the identical functional class. On the other hand, if links with the same functional class, but not in the forward direction, exist, then links with different functional classes may be considered as possible through links. So, the initial search is limited to an angle relative to exactly straight ahead, say to 60°. If this initial search fails to find a most likely through link, then the functional class and 60° angle restrictions are ignored and a second search is performed. It is possible that a most likely through link will not be found, in which case all of the subject link's lanes will be designated turning lanes.

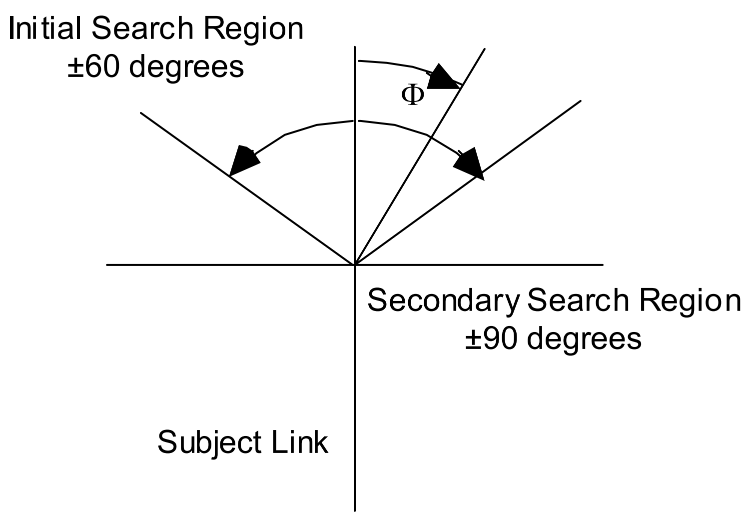

Fig. 2. Search for the most likely through link.

An additional exception in determining the through link exists when the subject link is a freeway ramp. In general, the through link is not expected to be another ramp, though there may be a ramp back onto the freeway located on the other side of the intersection. To force the algorithm to search for other possible through links onto which to route the vehicles, other ramps are excluded from being the through link in such situations. However, the outgoing ramp link is considered as either a left link or right link in the algorithm and connectivity from the "off" ramp to the "on" ramp is allowed.

1.6.2.5.4 Determine Through Lanes, Left Lanes, and Right Lanes

To track which links are left and right of the through link, we establish the following comparison function:

$$f\lambda = 2 - \cos(\Phi_\lambda) \qquad \text{if } x > 0,$$
$$f\lambda = \cos(\Phi_\lambda) \qquad \text{if } x \le 0,$$

where Φ_λ is the angle for link λ and x is the ordinate of the link's other node relative to the intersection node. As shown in Fig. 3, this function decreases monotonically counterclockwise from the subject link.

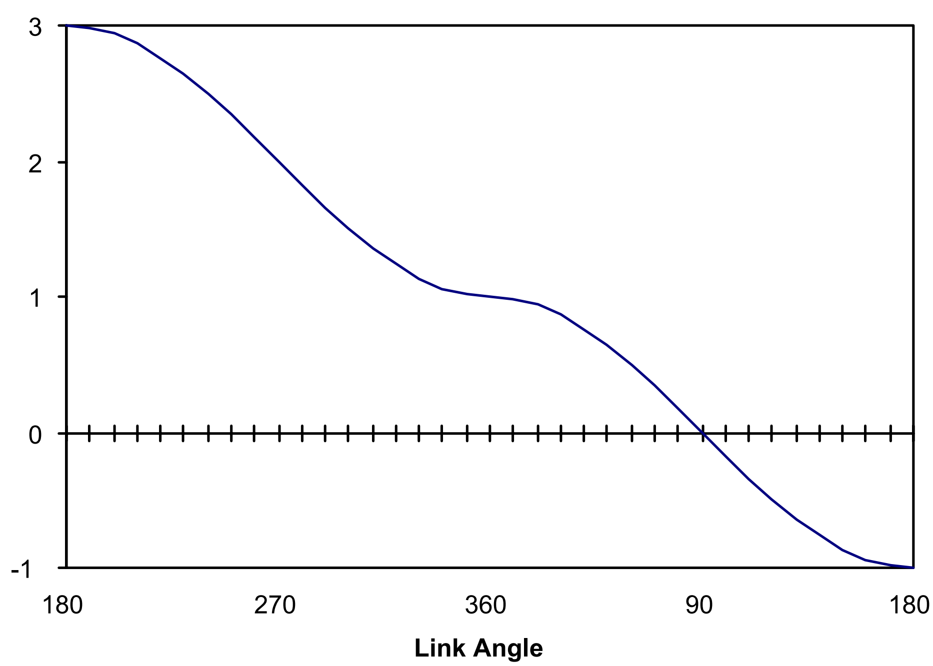

Fig. 3. Link comparison function.

Thus, for right-bound links r, $f_r > f_t$, and for left-bound links l, $f_l < f_t$, where t designates the through link. If there is no through link, then $f_t = 1$.

1.6.2.5.4.1 Through Lanes

The through-lane connectivities depend on the number of incoming permanent and turn pocket lanes on the subject link, the number of outgoing permanent and turn pocket lanes on the through link, and the presence of incoming and outgoing lanes, both left and right. The objective is to identify an equal number of incoming and outgoing through lanes on the subject and through links, respectively.

We are concerned about the presence of left outgoing lanes because if there are none, then the leftmost lane on the subject link is either a through lane, or possibly a right-turn lane if there also is no through link. Likewise, the presence of right outgoing lanes is cause for concern because their absence has connectivity implications for the rightmost lane. Thus, we make the following definitions:

- `LeftOutBound` is the number of outgoing permanent lanes on the left outgoing links, and
- `RightOutBound` is the number of outgoing permanent lanes on the right outgoing links.

Merge pockets are not considered in the current algorithm, so only the permanent outgoing lanes are counted.

Similarly, special attention must be given to the left and right incoming lanes, particularly if the through link has substantially more outgoing lanes than the subject link can feed as through lanes. The outgoing through lanes may be shifted left or right to accommodate the availability of the left and right incoming lanes that could turn onto the through link. (This algorithm does not account for prohibited left turns for which the connectivities have to be adjusted manually.) Again, we define:

- `LeftInBound` is the number of incoming lanes (including pocket lanes) on the left incoming links, and
- `RightInBound` is the number of incoming lanes (including pocket lanes) on the right incoming links.

We also define:

- `OutBound` is the number of outgoing permanent lanes on the through outgoing link,
- `InBound` is the number of permanent and turn pocket lanes on the subject link, and
- `LaneOffset` is the number of outgoing left-turn pocket lanes on the through link.

`LaneOffset` ensures that the through-link lane numbering accounts for the left-turn pocket lanes at the other through-link node (the through-link node away from the intersection for which we are building connectivities).

We initialize the limits of the outgoing through lanes,

```
FirstTOL = LaneOffset + 1,
LastTOL = OutBound + FirstTOL – 1,
```

where

- `FirstTOL` is the leftmost through outgoing lane on the through link, and
- `LastTOL` is the rightmost through outgoing lane on the through link.

Similarly, we determine the leftmost and rightmost incoming through lanes:

$$\mathtt{FirstTIL} = \mathtt{LPin} + 1 \qquad \text{if } \mathtt{LeftOutBound} > 0,$$

$$\mathtt{FirstTIL} = 1 \qquad \text{if } \mathtt{LeftOutBound} = 0,$$

$$\mathtt{LastTIL} = \mathtt{LPin} + \mathtt{LanesIn} \qquad \text{if } \mathtt{RightOutBound} > 0,$$

$$\mathtt{LastTIL} = \mathtt{LPin} + \mathtt{LanesIn} + \mathtt{RPin} \qquad \text{if } \mathtt{RightOutBound} = 0,$$

where

- `FirstTIL` is the leftmost through incoming lane on the subject link,
- `LastTIL` is the rightmost through incoming lane on the subject link,
- `LPin` is the number of incoming left pocket lanes on the subject link,
- `LanesIn` is the number of incoming permanent lanes on the subject link, and
- `RPin` is the number of incoming right pocket lanes on the subject link.

Fig. 4 shows a possible intersection configuration illustrating these definitions.

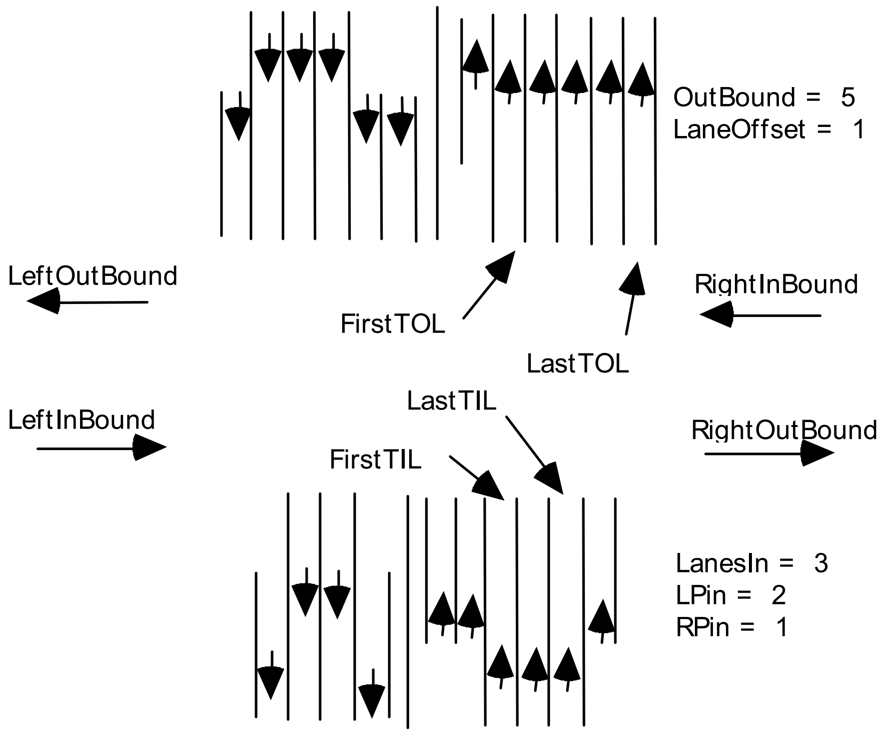

Fig. 4. Initial determination of through lanes.

In this scheme the permanent lanes are the only through lanes unless pocket lanes are denied any connectivity as determined by the availability of outbound lanes on the left and right. That is, if there are no outbound lanes on the left, then any incoming left pocket lanes must be considered as through lanes. Similar considerations apply when outbound lanes are not available on the right.

The number of possible through outbound lanes and the number of possible through inbound lanes is calculated:

$\texttt{ThruOutBound} = \texttt{LastTOL} - \texttt{FirstTOL} + 1$, and

$\texttt{ThruInBound} = \texttt{LastTIL} - \texttt{FirstTIL} + 1$.

To assure that the numbers of incoming and outgoing through lanes are equal, the smaller is chosen:

$\texttt{ThruBound} = \min(\texttt{ThruOutBound}, \texttt{ThruInBound})$.

Whether this leaves extra incoming lanes or extra outgoing lanes that will need to be changed from through lanes into turning lanes instead of through lanes is determined:

$\texttt{ThruExcess} = \texttt{ThruInBound} - \texttt{ThruOutBound}$.

If there are excess through lanes on the subject or through link, then the appropriate leftmost and rightmost through lane positions need to be adjusted to account for the excess. Also, the outbound right, left, and through lane availability must be calculated.

1.6.2.5.4.2 Excess Incoming Through Lanes

If `ThruExcess` is greater than zero, there are more incoming through lanes than outgoing through lanes. Therefore, the excess lanes must be redistributed and reassigned as left and right-turn lanes for the subject link. The excess lanes are divided alternately between left and right turns, beginning with right turns. There may already be left- and right-turn lanes located on the subject link,

$$\texttt{LTurnPrev} = \texttt{FirstTIL} - 1,$$

$$\texttt{RTurnPrev} = \texttt{InBound} - \texttt{LastTIL},$$

where `InBound` is the total number of lanes entering the intersection from the subject link. This situation is illustrated in Fig. 5.

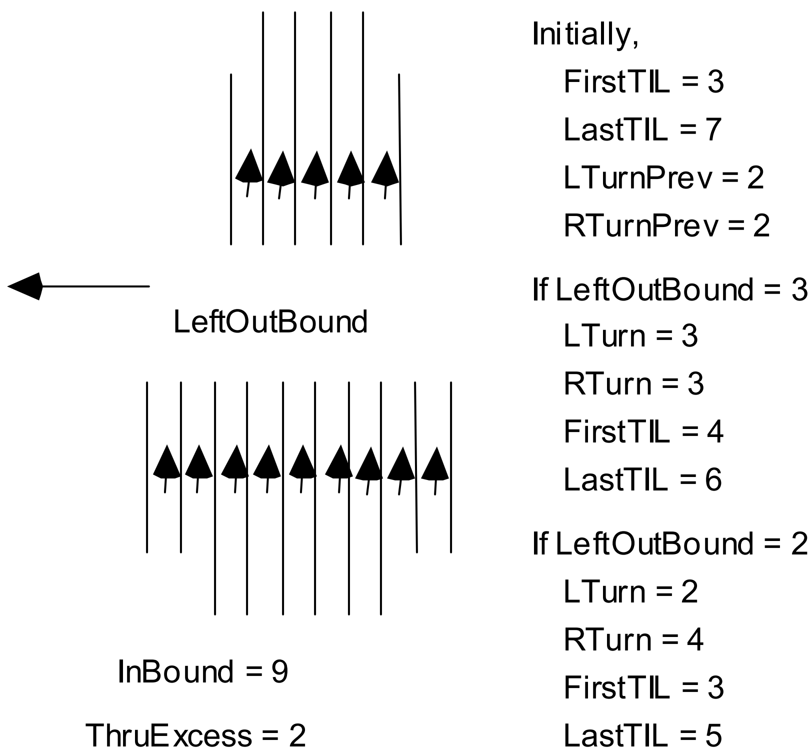

Fig. 5. Example of excess incoming through lanes.

The left outbound lanes, totaling `LeftOutBound`, may not be able to accommodate all of the left-turn lanes if half of the excess lanes are assigned as left-turn lanes. That is, half of the excess lanes are changed to left-turn lanes if, when added to `LTurnPrev`, the total is less than `LeftOutBound`; otherwise the total number of left-turn lanes is set to `LeftOutBound`,

$$\mathtt{LTurn} = \mathtt{LTurnPrev} + \mathrm{floor}(\mathtt{ThruExess}/2)$$
$$\text{if } \mathtt{ThruExcess}/2 \le (\mathtt{LeftOutBound} - \mathtt{LTurnPrev}),$$

$$\mathtt{LTurn} = \mathtt{LeftOutBound}$$
$$\text{if } \mathtt{ThruExcess}/2 > (\mathtt{LeftOutBound} - \mathtt{LTurnPrev}),$$

where the *floor* function returns the largest integer smaller than or equal to the value. At this point, the number of right-turn lanes is calculated,

$$\mathtt{RTurn} = \mathtt{InBound} - \mathtt{ThruBound} - \mathtt{LTurn},$$

and the through-lane leftmost and rightmost positions are updated,

$$\mathtt{FirstTIL} = \mathtt{LTurn} + 1,$$

$$\mathtt{LastTIL} = \mathtt{InBound} - \mathtt{RTurn}.$$

1.6.2.5.4.3 Excess Outgoing Through Lanes

When `ThruExcess` is less than zero, there are more outgoing than incoming through lanes and these excess lanes must be reassigned as through-link outgoing lanes connected to incoming lanes from the left or right links. Note that we do not assign the left or right link incoming connectivities at this point; that is done when the left or right links are rotated to become the subject link. However, by apportioning the excess lanes to the left and right on the through link, the positions of the leftmost and rightmost through lanes on the through link are adjusted.

The assignment of excess lanes is biased to the right so that the right outgoing lanes can (possibly) accept right-turning vehicles without interference from through traffic. First, it must be determined whether there are sufficient lanes inbound from the left and right to service the apportioned excess lanes. The minimum between the number of left and right inbound lanes is found and used to determine whether that minimum can service half of the excess lanes. If not, the number of excess lanes assigned to that side of the through lanes is limited.

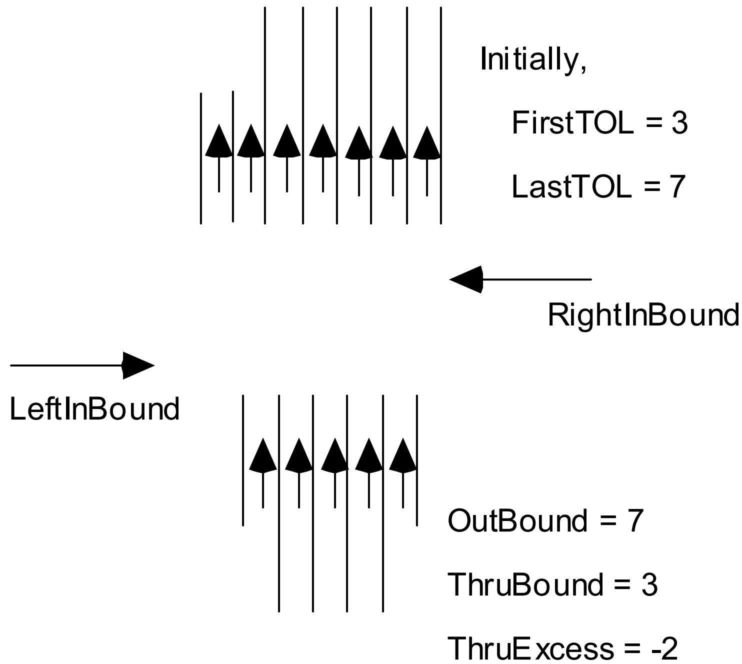

Fig. 6. Example of excess outgoing through lanes.

Using Fig. 6 as an illustrative example, when $\texttt{LeftInBound} > \texttt{RightInBound}$ and $\texttt{ThruExcess} > 2\,(\texttt{RightInBound} - \texttt{OutBound} + \texttt{LastTOL})$,

$$\texttt{LeftLanes} = \texttt{OutBound} - \texttt{ThruBound} - \texttt{RightInBound},$$

or, when $\texttt{LeftInBound} \le \texttt{RightInBound}$ and $\texttt{ThruExcess} > 2\,(\texttt{LeftInBound} - \texttt{FirstTOL} + 1)$,

$$\texttt{LeftLanes} = \texttt{LeftInBound};$$

otherwise,

$$\texttt{LeftLanes} = \texttt{FirstTOL} - 1 + \text{floor}(\texttt{ThruExcess}/2)$$

where $\texttt{LeftLanes}$ is the total number of through-link lanes that are connected to lanes from the left links, and the *floor* function returns the largest integer smaller than or equal to the value.

Now, the leftmost and rightmost outgoing through lanes are redefined,

$$\texttt{FirstTOL} = \texttt{LeftLanes} + 1,$$

$$\texttt{LastTOL} = \texttt{LeftLanes} + \texttt{ThruBound}.$$

1.6.2.5.4.4 Through-Lane Connectivities

At this point the lane connectivity table can be constructed. As shown in Fig. 7, for the through lanes the incoming lane `FirstTIL` of the subject link is connected with the outgoing lane `FirstTOL` of the through link, {`FirstTIL` + 1} with {`FirstTOL` + 1}, etc., until incoming lane `LastTIL` is connected to outgoing lane `LastTOL`.

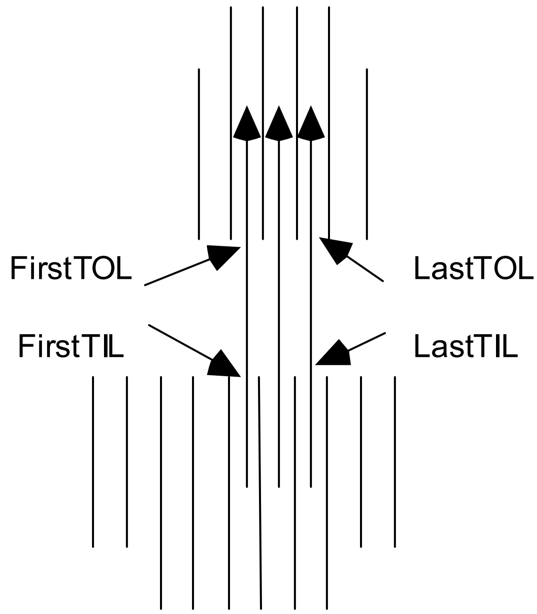

Fig. 7. Example of through lane connectivities.

1.6.2.5.4.5 Left-Turn-Lane Connectivities

Left-turn-lane connectivities are considered only if there are left outbound lanes. For the subject-link left-turn lanes, each left link is examined in a clockwise direction (the comparison function, f_λ, lets us order the links). First it is determined how many subject-link left-turn lanes must be assigned to the left link. All are assigned to this single left-turn link unless a second link exists, to which the remainder is assigned. As shown in Fig. 8, we know the total outbound left lanes, `LeftOutBound`, and the total outbound lanes on this left link,

`LOut` = `LPOut` + `LanesOut` + `RPOut`,

where `LPOut`, `LanesOut`, and `RPOut` are defined for this left link using the same procedure as outlined earlier for determining through lanes. The difference between `LeftOutBound` and `LOut` is the number of lanes on additional left links available for left-turns from the subject link. So, from the `LTurn` left lanes, at least (`LTurn` – (`LeftOutBound` – `LOut`)) must be used on this left link.

Also, any outgoing left pocket merge lanes must be found since they will be utilized first. So,

$\texttt{OutLane} = \texttt{LPOut}$

$\quad$ if $\texttt{LPOut} >= (\texttt{LTurn} - (\texttt{LeftOutBound} - \texttt{LOut}))$,

$\texttt{OutLane} = \texttt{LTurn} - (\texttt{LeftOutBound} - \texttt{LOut})$,

$\quad$ if $\texttt{LPOut} < (\texttt{LTurn} - (\texttt{LeftOutBound} - \texttt{LOut}))$,

though,

$\texttt{OutLane} = 1$, if the previous result is less than or equal zero,

where `OutLane` is the number of lanes on this left link to be connected to left turns from the subject link. Now each subject link's left-turn lane, starting at lane number one, is connected to each left link's lane, also starting at lane number one, incrementing up to `OutLane`.

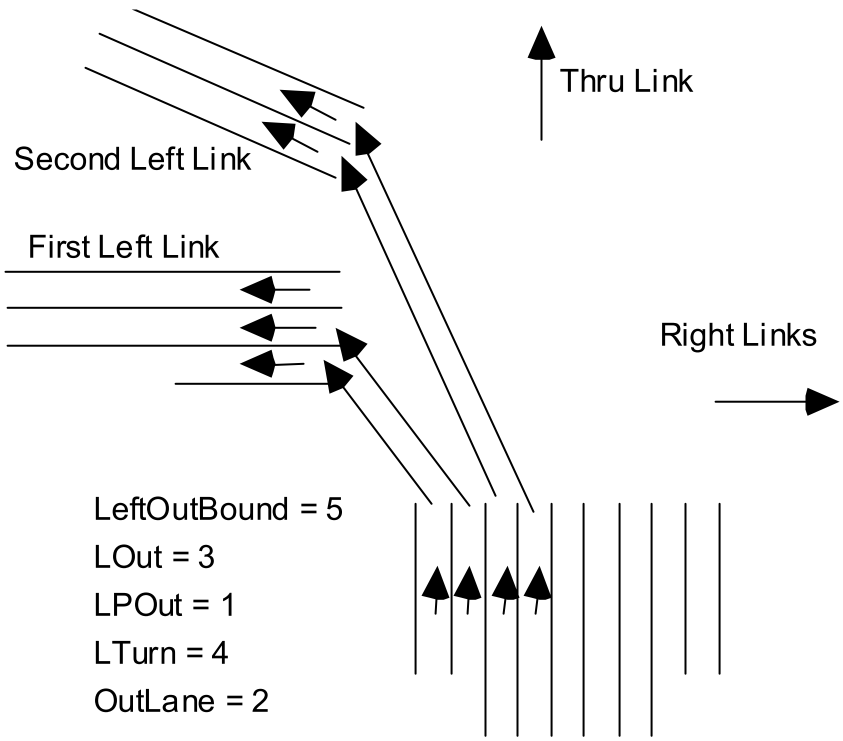

Fig. 8. Example of left-turn lane connectivities.

If there is another left link, the same procedure is applied, although `LeftOutBound` must be updated to reflect that the lanes of previous left links are no longer available, and `LTurn` must be reduced by `OutLane` because those lanes have already been utilized. Each subject link's left-turn lane must be connected (starting, not from 1, but from `OutLane` + 1, the next lane over from where we stopped on the previous left link).

However, the subject link's left-turn lane cannot to exceed `FirstTIL`, the first through lane. This is avoided by always starting from lane number one.

This process is continued for each left link. If all the subject link's left-turn lanes have been used, and there are additional left links with outgoing lanes, the left turns from the subject link's first through lane, `FirstTIL`, are assigned to the first lane, lane number one, of the additional left links.

1.6.2.5.4.6 Right-Turn-Lane Connectivities

As illustrated in Fig. 9, the right-turn-lane connectivities are assigned in the same manner previously described for the left-turn lanes; we continue to scan the links in a clockwise direction and to assign the subject link's lanes from left to right. As a result, after all the right-turn lanes have been accounted for, any necessary additional right-turns from the subject link's far right lane are created. As before, the total outbound right lanes, `RightOutBound`, and the total outbound lanes on this right link,

$$\texttt{ROut} = \texttt{LPOut} + \texttt{LanesOut} + \texttt{RPOut},$$

are known.

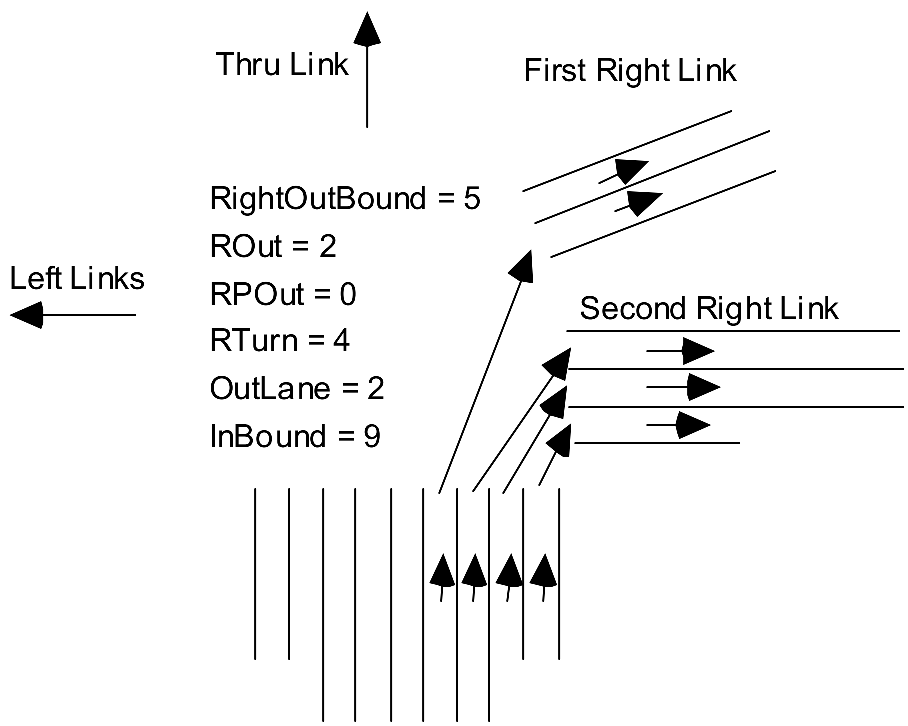

Fig. 9. Example of right-turn lane connectivities.

Using the same arguments outlined for the left-turn-lane connectivities, but applying them to the right lanes, yields

$$\texttt{OutLane} = \texttt{ROut} - \texttt{RPOut} + 1,$$

$$\text{if } \texttt{RPOut} >= (\texttt{RTurn} - (\texttt{RightOutBound} - \texttt{ROut})),$$

$$\texttt{OutLane} = \texttt{RightOutBound} - \texttt{RTurn} + 1,$$

$$\text{if } \texttt{RPOut} < (\texttt{RTurn} - (\texttt{RightOutBound} - \texttt{ROut})),$$

though

$$\texttt{OutLane} = \texttt{ROut}, \qquad \text{if the previous result is greater than } \texttt{ROut}.$$

Beginning with lane $\{\texttt{InBound} - \texttt{RTurn} + 1\}$ on the subject link and lane `OutLane` on the right link, the connectivity between the lanes, incrementing up to `Rout`, is assigned. For each additional link, the same procedure is followed, updating `RightOutBound` and `RTurn`. The process is continued for each right link. If all of the subject link's right-turn lanes have been utilized and there are additional right links with outgoing lanes, the right turns from the subject link's last lane, `InBound`, are assigned to the last lane, `ROut`, of the additional right links.

At this point the lane connectivities for the subject link are complete, and the intersection is rotated to select another subject link; or if the intersection is complete, the next intersection is selected and this process is repeated.

1.6.3 CreateTrafficControls

1.6.3.1 Overview

CreateTrafficControls is a program that creates a traffic control for all network nodes as required by the TRANSIMS microsimulation. The type of control may be a traffic signal, a traffic sign, or uncontrolled. For a signalized control, the program creates a fully actuated signal with appropriate phasing and timing plans and associated traffic detectors.

The *CreateTrafficControls* tool may be used in the absence of any traffic control information to generate controls at all nodes. It may also be used in conjunction with existing traffic control information and will then generate controls for only those nodes that still require a control. When used in this mode, existing files and generated files must be manually concatenated before being used as input to the applications.

The program implements several sets of heuristics for determining traffic control characteristics. First, characteristics of the links around an intersection are used to decide whether the intersection will be signalized or unsignalized. For a signalized intersection, the allowed movements at the intersection as specified by the lane connectivity and the geometry of the intersection are used to determine the required phases in the phasing plan and to specify the permitted movements within each phase. For an unsignalized intersection, the link characteristics are used to determine the type of sign control. Nodes that do not occur at intersections are treated as unsignalized with no control on the links. Depending on the precise link characteristics, a small number of nodes—typically at the periphery of the simulated region—may not receive any type of control. These nodes are said to have "null" control. Simulated traffic may not enter a node with a null control and valid plans will not attempt this.

Other network tools may need to be run prior to running *CreateTrafficControls*. If a lane connectivity table does not already exist, the *LaneConnectivity* program may be used to generate this table. *CreateTrafficControls* relies heavily on accurate information in the node, link, pocket lane, and lane connectivity tables. Invalid information may cause spurious traffic controls to be created. It is strongly recommended to run the *ValidateNetwork* program and correct identified network errors in these tables prior to running *CreateTrafficControls*. Additionally, *CreateTrafficControls* is equipped with several warning and error messages that should be investigated if they occur. This may lead to the discovery of errors in existing tables and the iterative refinement of several tables.

1.6.3.1.1 Usage

```
CreateTrafficControls <configuration file> <log file>
```

1.6.3.2 Configuration File Keys

Valid values for the following network configuration keys must be provided in the configuration file.

```
NET_DIRECTORY

NET_LANE_CONNECTIVITY_TABLE

NET_LINK_TABLE

NET_NODE_TABLE

NET_POCKET_LANE_TABLE
```

If the following configuration file keys specify existing data files, the information will be used by *CreateTrafficControls*. Otherwise, the configuration file keys should specify empty data files with proper headers.

```
NET_DETECTOR_TABLE

NET_PHASING_PLAN_TABLE

NET_SIGNALIZED_NODE_TABLE

NET_TIMING_PLAN_TABLE

NET_UNSIGNALIZED_NODE_TABLE
```

NET_SIGNAL_COORDINATOR_TABLE

The following configuration file keys should specify empty data files with proper headers.

NET_ACTIVITY_LOCATION_TABLE

NET_BARRIER_TABLE

NET_LANE_USE_TABLE

NET_PARKING_TABLE

NET_PROCESS_LINK_TABLE

NET_SPEED_TABLE

NET_TRANSIT_STOP_TABLE

NET_TURN_PROHIBITION_TABLE

Additional configuration file keys used only by *CreateTrafficControls* are

defined in Table 45. Note that the algorithms are described in more detail below.

Table 45. Configuration file keys used only by *CreateTrafficControls*.

Configuration File Key	Description
NET_CONTROL_GENERATOR_ALGORITHM	The algorithm to be used to determine whether a node requires a signalized or unsignalized control. Valid values are F or L. No default value.
NET_CONTROL_GENERATOR_SINGLE_NODE	Indicates whether traffic controls are created for only one node. This may be useful for testing. A valid value is a node ID.
NET_CONTROL_GENERATOR_WARNINGS	Indicates whether warning messages will be printed. Valid values are true or false. Default = true
NET_DETECTOR_LENGTH	The standard length of a detector in meters. When a link is shorter than this length, the detector is as long as the link. Detectors for turn pockets are as long as the pocket. Default length = 37.5 meters
NET_DETECTOR_PER_LANE	Indicates whether a detector is created for every lane rather than spanning multiple lanes. Valid values are true or false. Default = false

Configuration File Key	Description
NET_LANES_COUNT	A parameter for algorithm L. Indicates the number of lanes on a link that will cause a signalized control to be generated. Default = 3
NET_LOCAL_LANES_COUNT	An optional parameter for algorithm F. Indicates the number of lanes on a local street that will cause a signalized control to be generated. Default = 99
NET_NEW_DETECTOR_TABLE	The name of the detector table that will be created. No default value.
NET_NEW_PHASING_PLAN_TABLE	The name of the phasing plan table that will be created. No default value.
NET_NEW_SIGNAL_COORDINATOR_TABLE	The name of the coordinator table that will be created. No default value.
NET_NEW_SIGNALIZED_NODE_TABLE	The name of the signalized node table that will be created. No default value.
NET_NEW_TIMING_PLAN_TABLE	The name of the timing plan table that will be created. No default value.
NET_NEW_UNSIGNALIZED_NODE_TABLE	The name of the unsignalized node table that will be created. No default value.
NET_TIMING_CYCLE_LENGTH	The length (in seconds) of the timing cycle. Default = 60
NET_TIMING_GREENEXT_FRACTION	The GREENEXT parameter will be assigned a value that is a fraction of the computed GREENMIN value. Default = 0.6
NET_TIMING_GREENMIN_MINIMUM	The minimum value (in seconds) that the GREENMIN parameter may assume. Default = 5
NET_TIMING_GREENMIN_TOLERANCE	Two timing plans will be considered the same if the GREENMIN values for all phases are within this tolerance. Default value = 0.0001 (i.e., almost no tolerance)
NET_TIMING_REDCLEAR	The length (in seconds) of the red clearance interval. Default = 1
NET_TIMING_YELLOW	The length (in seconds) of the yellow interval. Default = 3.

1.6.3.3 Log File Output

The log file may contain error messages and, optionally, warning messages (see Table 46). Error messages must be addressed and the output tables corrected. Warning messages often indicate network problems that should be corrected.

Table 46. Error and warning messages.

Error 19012	Node "nodeid" has link "linkid" with functional class "f". No control assigned.
Solution	This error message occurs when a link at an unsignalized intersection has one of the following functional classes: bikeway, busway, heavy rail, ferry, zonal connector. These classes of roadway are not handled by *CreateTrafficControls* and the link is defined as being uncontrolled. All such links should be manually examined to determine the appropriate type of control and the output should be edited accordingly.
Error 19013	Node "nodeid" link "linkid" to link "linkid" not in any phase.
Solution	This error message occurs when a movement allowed in the lane connectivity table was not assigned to any phase. This intersection must be examined and corrected manually.
Error 19014	Node "nodeid" link "linkid" has lanes not assigned.
Solution	This error message occurs when some lanes for a link at a signalized intersection were not assigned to a phase. This intersection must be examined and corrected manually.
Warning 19015	Node "nodeid" (link "linkid") has in degree = "d1" and out degree = "d2".
Solution	This warning message occurs when the number of incoming links at a node does not equal the number of outgoing links. This is often not a problem, but nodes with incoming degree = 0 or outgoing degree = 0 are likely erroneous and should be examined.
Warning 19016	Node "nodeid" (link "linkid") has functional classes: "f".
Solution	This warning message occurs when the incoming links at a node have more than two distinct functional classes. This is not necessarily a problem, but can reveal unusual cases.
Warning 19017	Node "nodeid" link "linkid" respecifies straight link "linkid".

Solution	For each incoming link at a node, the algorithm tries to determine whether there is another incoming link in the straight direction. The tolerance for determining the straight direction is initially 20 degrees. If more than one incoming link finds the same link to be straight ahead, the tolerance is iteratively halved in an attempt to narrow to a single link. This warning occurs when more than one incoming link at a node specifies the same link as being straight ahead after four iterations. This may not be a problem, but indicates a very tight angle between two links at the node or that two links completely overlap. This case is suspicious and should be examined.
`Warning 19018`	`Node "nodeid" (link "linkid") has in degree = "d".`
Solution	This warning message occurs when the number of incoming links at a node is greater than 4. This may not be a problem, but nodes with a high degree are suspicious and should be examined.
`Warning 19019`	`Node "nodeid" (link "linkid") has in degree = "d" and null control.`
Solution	This warning message occurs when there are no incoming links at a node or when there is one incoming link with no connectivity to an outgoing link. This node will have a "null" control in the network database. These nodes are suspicious and should be examined and corrected if they do not occur on the periphery of the network.
`Warning 19020`	`Node "nodeid" has link "linkid" with functional class OTHER. Stop sign assigned.`
Solution	This warning message occurs when a link at an unsignalized node has the OTHER functional class. This class is typically used when none of the other classes is appropriate, and these links should be examined to determine whether a stop sign is the appropriate control.
`Warning 19021`	`Node "nodeid" has link "linkid" with functional class LIGHTRAIL.`
Solution	This warning message occurs when a link at a signalized or unsignalized node has the light rail functional class. These links should be examined to determine whether no control is appropriate for an unsignalized node, or whether the phasing at a signalized node is correct.

1.6.3.3.1 Phasing Patterns

Following any error or warning messages produced while traffic controls are generated, the log file contains a synopsis of the symbolic phasing patterns that were produced. The meaning of the components of the patterns is described in Section 2.5. The first column lists the pattern followed by a list of the nodes that were assigned a control with this pattern. This list should be checked for suspicious patterns and nodes with those patterns should be examined and possibly modified.

Some examples of suspicious patterns are:

- patterns with a single phase (e.g., A or C only)
- patterns such as AAA
- patterns with more than 4 phases

1.6.3.3.2 Summary of Counts of Generated Controls

The final log output is a summary of the counts of each type of control that was generated. When no existing traffic control tables were used as input, the sum should equal the total number of nodes. When existing tables were used, the sum should equal the total number of nodes minus the nodes in the existing tables. Any discrepancies in the counts should be investigated.

1.6.3.4 Traffic Control Table Output

The following five traffic control tables specified by the network representation are produced by *CreateTrafficControls*. Each of these files must be manually concatenated with existing network files of the same type that were used as input by *CreateTrafficControls*. Detector IDs and timing plan IDs that are generated by this program will be larger than IDs in any existing files.

1.6.3.4.1 Signalized Node Table

The signalized node table is written to the file named by the `NET_NEW_SIGNALIZED_NODE_TABLE` configuration file key. All nodes in this file specify a single ring-actuated control using actuated algorithm B. The `NOTES` field contains the symbolic phasing pattern generated at the node.

1.6.3.4.2 Phasing Plan Table

A complete phasing plan for each signalized node is written to the file named by the `NET_NEW_PHASING_PLAN_TABLE` configuration file key.

1.6.3.4.3 Timing Plan Table

Timing plans are created in the file named by the NET_NEW_TIMING_PLAN_TABLE configuration file key. The interval lengths are determined by the NET_TIMING* configuration file keys and an algorithm described below. All potential phases are included in the NEXTPHASES field.

1.6.3.4.4 Detector Table

The detector table is written to the file named by the NET_NEW_DETECTOR_TABLE configuration file key. Presence detectors with a defect category of 0 (no defects) are defined for each incoming link at a signalized node. The detector offset is at the link's setback, and the detector length is specified by the NET_DETECTOR_LENGTH configuration file key, except for turn pocket lanes where the detector length is the length of the pocket. A detector spans as many lanes as possible for the type of allowed movement unless NET_DETECTOR_PER_LANE is specified true.

1.6.3.4.5 Unsignalized Node Table

The unsignalized node table is written to the file named by the NET_NEW_UNSIGNALIZED_NODE_TABLE configuration file key. A record is written for each incoming link at an unsignalized node.

1.6.3.4.6 Signal Coordinator Table

The signal coordinator table is written to the file names by the NET_NEW_SIGNAL_COORDINATOR_TABLE configuration file key. An isolated control coordinator record is written for each signalized node.

1.6.3.5 Symbolic Phasing Patterns

For purposes of generating phasing plan records at a signalized node, the topological characteristics at the node are abstracted into three general patterns.

1) Pattern A represents the case of unprotected movements on two links that are opposite each other. This is the typical through movement case where unprotected left (and right) turns are also allowed. The visual signal control is a green ball.

2) Pattern B represents the case of protected left turn movements on two links that are opposite each other. The visual signal control is a green left arrow.

3) Pattern C represents the case of protected movements in all directions from a single link. This case occurs when there is no straight through link or when this link has a left-turn pocket but the opposite link does not. The visual signal control is green arrows in every direction.

1.6.3.6 Algorithms

Two algorithms are provided for determining whether the traffic control to be generated is signalized or unsignalized. The desired algorithm is selected with the `NET_CONTROL_GENERATOR_ALGORITHM` configuration file key.

Both algorithms generate an unsignalized control when the incoming degree at the node is 2 or less. Both algorithms generate a signalized control if any incoming link has a left turn pocket. Algorithmic differences are described in the following two sections.

1.6.3.6.1 Algorithm L

With algorithm L, if any incoming link has at least `NET_LANES_COUNT` lanes, a signalized control is generated at the node. Otherwise, the node is unsignalized.

1.6.3.6.2 Algorithm F

Algorithm F uses the functional classes of incoming links to determine the type of control to be generated. The intersection of a ramp and any other type of street is unsignalized. The intersection of a local street and any other type of street is unsignalized, unless the local street has at least `NET_LOCAL_LANES_COUNT` lanes, in which case the intersection is signalized. The intersection of an expressway, primary arterial, secondary arterial, collector, or frontage road and any other type of street is signalized.

1.6.3.6.3 Phasing Algorithm

After it is determined that a signalized control will be generated, the phasing for the signal must be determined. The algorithm iterates through the incoming links proceeding as follows:

- If there is another link straight opposite this link and if both this link and that link have left-turn-only lanes, a phase with pattern B is created.
- If only this link has left-turn-only lanes, a phase with pattern C is created.
- If there is another link straight opposite this link, a phase with pattern A is created.
- If there is no link straight opposite this link, a phase with pattern C is created.

The tolerance for determining whether a straight opposite link exists is defined to be 20 degrees. Links are marked so that patterns A and B are generated only once for each opposing pair of links. Phasing records are generated for each allowed movement during the phase, including "unprotected after stop" movements.

1.6.3.6.4 Sign Control Algorithm

When one of the above algorithms determines that an unsignalized control will be generated, it remains to be determined whether each incoming link will have a stop sign, a yield sign, or no sign control. The algorithm proceeds as follows:

- If the incoming degree at the node is 2, no sign control is assigned to each incoming link.
- If the incoming degree is greater than 2, a yield sign is assigned to ramps, a stop sign is assigned to local streets, and all other types of streets receive no sign control.
- If the incoming degree is less than 2 and there is no connectivity out of the node, the node is left uncontrolled and a null control is assigned during subsequent network constructions. A warning is printed when this case occurs.

1.6.3.6.5 Timing Algorithm

The minimum green length for a phase is based on a weighted sum of the number of incoming lanes participating in the phases. A permanent lane has a weight of 1, and a pocket lane has a weight of 1/3. The total cycle length is apportioned to the phases with constant yellow and red clearance intervals, and minimum green intervals based upon the ratio of the number of incoming lanes in this phase to the total number of lanes. The minimum value must be at least `NET_TIMING_GREENMIN_MINIMUM`, which may sometimes cause the cycle length to exceed `NET_TIMING_CYCLE_LENGTH`.

2. Transit Files

In this section, we discuss how to describe transit routes and schedules for the TRANSIMS Route Planner and Traffic Microsimulator. Before reading the sections in this volume, you should have already read Volume One (*Technical Overview*). This volume serves as a foundation for understanding Volume Three (*Modules*).

2.1 File Format

The transit route file (configuration file key `TRANSIT_ROUTE_FILE`) describes the transit network topology. This ASCII text file describes the transit stops along which vehicles on each transit route are allowed to stop.

The transit schedule file (configuration file key `TRANSIT_SCHEDULE_FILE`) describes the transit schedule. This ASCII text file lists the times at which a transit vehicle visits the transit stops.

The transit zone file (configuration file key `TRANSIT_ZONE_FILE`) describes transit fares. This ASCII text file lists the costs associated with riding a transit vehicle between transit fare zones.

2.1.1 Transit Route File Format

An ASCII text file, the Transit Route File has fields that are separated by white space [e.g., space(s), tab(s), or newline(s)]. The column names are currently not part of the route files. Table 47 lists the transit route file data definitions and format.

Table 47. Transit route file data definitions and format.

Column Name	Description	Allowed Values
`Transit Route ID`	A unique identifier for this route.	integer
`Number of Stops`	The number of transit stops to follow.	integer

`Transit Type`	The type of transit vehicle serving this route.	`BUS`
		`TROLLEY`
		`STREETCAR`
		`LIGHTRAIL`
		`RAPIDRAIL`
		`REGIONALRAIL`

`Transit Stop ID`	The ID of the transit stop.	integer
`Link ID`	The ID of the link on which the transit stop resides.	integer
`Node ID`	The ID of the node toward which the vehicle is heading.	integer
`Transit Zone`	The ID of the zone in which the transit stop is located (or 0 if the cost is not zone based).	integer

For each route, the file contains:

- the route ID,
- the number of transit stops the route visits, and
- the type of transit vehicle serving this route.

The route ID must be unique across transit types. Each route record is followed by multiple transit stop records (as many as is specified by the number of stops found in the route record). The transit stop IDs must be unique within a particular route.

For each stop, the following are listed:

- the transit stop ID (in the order visited by the transit vehicle),
- the link on which the stop resides,
- the node the vehicle is heading toward, and
- the transit zone in which the stop is located.

The zone field is 0 if the zones are not used for determining rider costs.

2.1.2 Transit Schedule and Format

The Transit Schedule File is an ASCII text file whose fields are separated by white space (as described above). The file must be sorted by Transit Route ID and time—in that order. The column names are currently not part of the schedule file. Table 48 provides transit schedule file data definitions and format.

Table 48. Transit schedule file data definitions and format.

Column Name	Description	Allowed Values
`Transit Route ID`	A unique identifier for this route.	integer
`Time`	The departure time at the stop.	integer: seconds since midnight
`Transit Stop ID`	The ID of this transit stop, as specified in the network data tables.	integer

2.1.3 Transit Zone File Format

The Transit Zone File is an ASCII file whose fields are separated by white space. This file describes the costs for traveling between zones on particular types of transit. The zones correspond to the zones given in the route file. The column names are currently not part of the zone file. The zone file is not currently used by TRANSIMS. Table 49 provides a list of transit zone file data definitions and format.

Table 49. Transit zone file data definitions and format.

Column Name	Description	Allowed Values
`From Zone`	The zone in which the transit trip begins	integer
`To Zone`	The zone in which the transit trip ends.	integer

`Transit Type`	The type of transit to which the `cost` is applicable	`BUS`
		`TROLLEY`
		`STREETCAR`
		`LIGHTRAIL`
		`RAPIDRAIL`
		`REGIONALRAIL`

`Cost`	The cost of the transit trip, in cents	integer

2.2 Files

Table 50 lists the transit library files.

Table 50. Transit library files.

Type	File Name	Description
Binary Files	*libTIO.a*	The TRANSIMS Interfaces library.
Source Files	*transitio.c*	The transit data structures and interface functions file.
	transitio.h	The transit interface functions source file.

2.3 Configuration File Keys

Table 51 lists the transit configuration file keys.

Table 51. Transit file configuration file keys.

Configuration File Key	Description
TRANSIT_ROUTE_FILE	The name of a transit route file whose format is described. Used as input by the Traffic Microsimulator and the Route Planner.
TRANSIT_SCHEDULE_FILE	The name of a transit schedule file whose format is described above. Used as input by the Route Planner.
TRANSIT_ZONE_FILE	The name of a transit zone file whose format is described above.

3. Vehicle File

This section describes the TRANSIMS vehicle file. The TRANSIMS Population Synthesizer generates and assigns private vehicles to households. The Activity Generator assigns a set of possible vehicles to each member of a household.

Freight and transit vehicles (and the plans for their drivers) are generated by separate utilities, but these must be included in the vehicle database. The vehicle IDs assigned by these utilities must be unique.

3.1 File Format

Fields in the ASCII vehicle file are tab- or space-delimited. The first line of the file must contain the field names. Each line of the vehicle file contains five mandatory fields:

1) the household ID,
2) the vehicle ID,
3) the ID of the starting location,
4) the type of the vehicle, and
5) a user-defined vehicle subtype used for emissions.

Each line may contain optional integer fields whose meaning is user defined. The number of these identifier fields may vary among different vehicle files. The number of optional identifier fields must be the same on every line within a vehicle file. The value *-1* is used as a default placeholder value for both the starting location and optional integer fields when the values are unknown or unused. Table 52 specifies the content of the required fields in the vehicle file.

Table 52. Vehicle file specification.

Column Name	Description
HHID	The household ID.
VEHICLE	The vehicle ID.
LOCATION	The ID of vehicle's starting location.
VEHTYPE	The vehicle type, which is one of the following: 1 = AUTO 2 = TRUCK 4 = TAXI 5 = BUS 6 = TROLLEY 7 = STREETCAR 8 = LIGHTRAIL 9 = RAPIDRAIL 10 = REGIONALRAIL
VSUBTYPE	The vehicle subtype, used for emissions. The vehicle subtype must correspond to a vehicle subtype specified in the vehicle prototype file.

3.2 Files

Table 53 contains vehicle library files

Table 53. Vehicle library files.

Type	File Name	Description
Binary Files	*libTIO.a*	The TRANSIMS Interfaces library.
Source Files	*vehio.h*	The vehicle data structures and interface functions file.
	vehio.c	The vehicle interface functions source file.

3.3 Configuration File Keys

Table 54 contains vehicle file configuration file keys.

Table 54. Vehicle file configuration file key.

Configuration File Key	Description
VEHICLE_FILE	The path of the vehicle file.

3.4 Examples

Appendix B contains vehicle file examples.

4. Vehicle Prototype File

This section describes the TRANSIMS vehicle prototype file. This file is used primarily by the Traffic Microsimulator to define characteristics common to whole categories of simulated vehicles. These characteristics include the vehicle type and subtype, its maximum velocity and acceleration, the vehicle length, and its occupant capacity. All vehicles used in the simulation must have a prototype.

4.1 File Format

Fields in the vehicle prototype file are tab- or space-delimited. The first line of the file must contain the field names.

Each line of the vehicle prototype file contains six mandatory fields:

1) the vehicle type,
2) the user-defined vehicle subtype used for emissions,
3) the vehicle maximum velocity,
4) the vehicle maximum acceleration,
5) the vehicle length, and
6) the vehicle capacity.

Table 55 describes the contents of the vehicle prototype files.

Table 55. Vehicle prototype file specification.

Column Name	Description
VEHTYPE	The vehicle type, which is one of the following: 1 = AUTO 2 = TRUCK 4 = TAXI 5 = BUS 6 = TROLLEY 7 = STREETCAR 8 = LIGHTRAIL 9 = RAPIDRAIL 10 = REGIONALRAIL
VSUBTYPE	The vehicle subtype, used for emissions.
MAXVEL	The maximum velocity (meters/second).
MAXACCEL	The maximum acceleration (meters/second/second).
LENGTH	The vehicle length (meters).
CAPACITY	The vehicle capacity (driver + number of possible passengers).

4.2 Files

Table 56 contains vehicle prototype library files.

Table 56. Vehicle prototype library files.

Type	File Name	Description
Binary Files	*libTIO.a*	The TRANSIMS Interfaces library.
Source Files	*vehprotoio.h*	The vehicle prototype data structures and interface functions file.
	vehprotoio.c	The vehicle prototype functions source file.

4.3 Configuration File Keys

Table 57 shows the vehicle prototype configuration file key.

Table 57. Prototype file configuration file key.

Configuration File Key	Description
`VEHICLE_PROTOTYPE_FILE`	The path of the vehicle prototype file.

4.4 Examples

Appendix C contains vehicle prototype file examples.

Appendix A: Network Files Examples

Fig. 10 shows the layout of the network used for testing various TRANSIMS modules. Note the following about this network:

- Three of the nodes (8521, 14136, and 14141) in the node are associated with intersections, while another (8520) is associated with a change in the permanent number of lanes.
- Entries in the link table illustrate how the permanent number of lanes, left pocket lanes, and right pocket lanes are counted and numbered.
- All three types of pocket lanes (turn pocket, merge pocket, and pull-out pocket) are represented in the pocket lane table.
- Several types of parking (lot, street, driveway, and generic vs. actual) are represented in the parking table.
- The first six rows in the lane connectivity table may be understood as follows: Lanes 1 and 2 on link 11487 (attached to node 14141) are exclusive left-turn lanes connecting only to lanes 1 and 2 on link 11486. Lanes 3, 4, and 5 on link 11487 are through lanes to lanes 1, 2, 3, and on link 11495. Lane 6 on link 11487 is a right-turn-only lane connecting to lane 3 on link 28800. Lane 3 on link 11486 connects on both lane 2 on link 28800 and lane 3 on link 11487, as shown in rows 9 and 10 of the table.
- The number of permanent lane changes from three lanes changes from three lanes on link 28800 to four lanes on link 12384 at node 8520. No right-of-way sign control is required at this node. A stop sign is indicated on link 12407 at node 14136, with no sign control on the other two links at this node. The unsignalized node table illustrates this.
- Node 14141 has a pre-timed signal control with an offset of 19.0 seconds. A single timing and phasing plan is always in effect. Node 8521 is defined as having an actuated signal and two timing and phasing plans. The signalized node table illustrates this.
- The movements permitted during phase 1 at node 14141 are through movements between links 11487 and 11495, as well as right-turn movements from these links. Additionally, unprotected right turns are permitted from links 11486 and 28800 during phase 1. The first six rows of the phasing plan table specify this information.
- Plan 1 in the timing table was specified in the signalized node table as acceptable to node 14141. This is a timed signal with green, yellow, and red clearance intervals as indicated in row 1 of the table. Plans 2 and 3 for node 8521 were invented as illustrations and may not make sense as real timing plans.

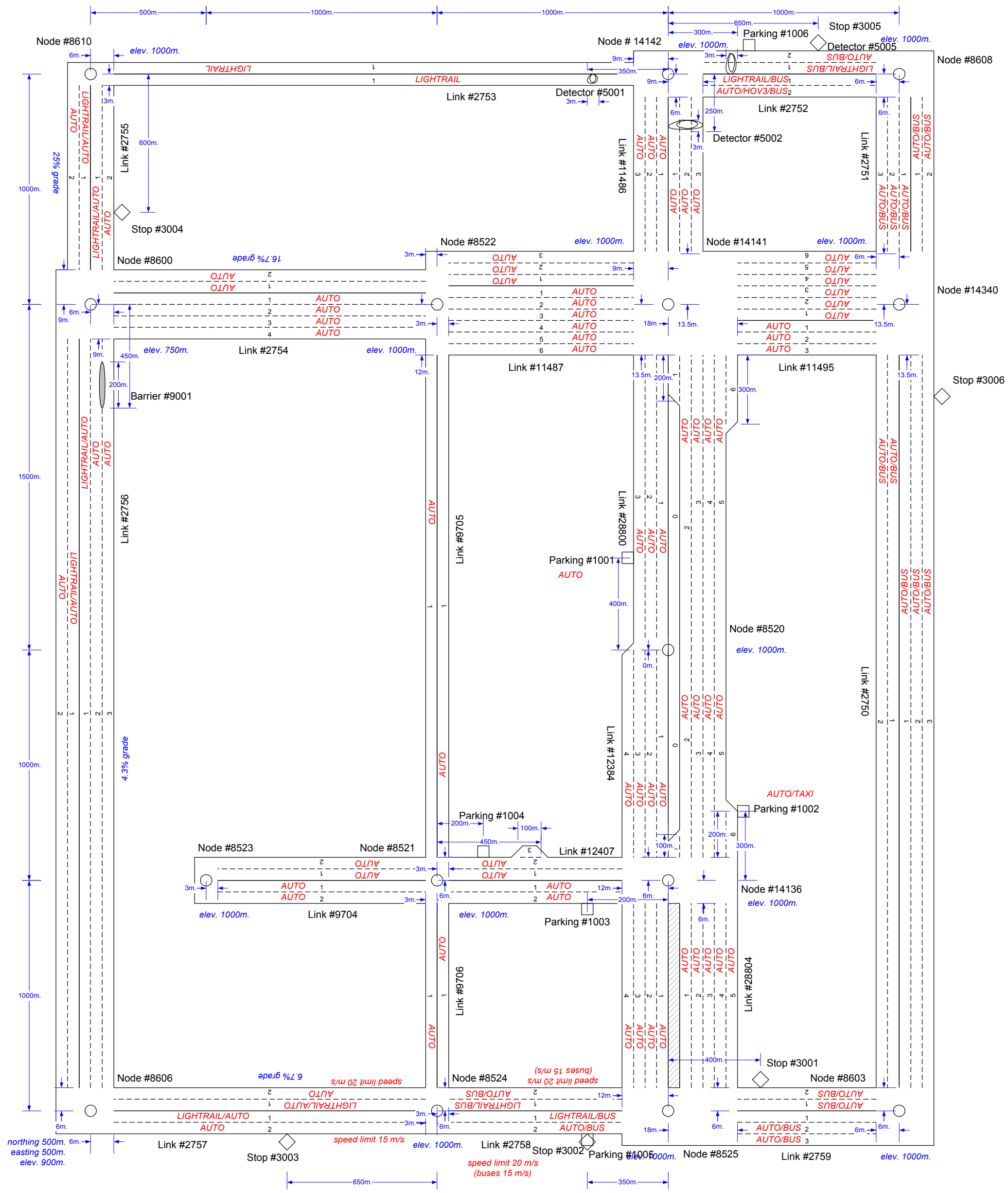

Fig. 10. Layout of test network.

The following is the configuration file for the network. This network contains most of the network objects available in TRANSIMS. Refer back to Table 22 for a listing the configuration file keys for this network.

```
# Network subsystem configuration file keys for the test network.

# The directory where the network files reside.
NET_DIRECTORY /home/projects/transims/networks/test

# The node table name.
NET_NODE_TABLE Test_Node_Table

# The link table name.
NET_LINK_TABLE Test_Link_Table

# The pocket lane table name.
NET_POCKET_LANE_TABLE Test_Pocket_Lane_Table

# The parking table name
NET_PARKING_TABLE Test_Parking_Table

# The lane connectivity table name.
NET_LANE_CONNECTIVITY_TABLE Test_Lane_Connectivity_Table

# The unsignalized node table name.
NET_UNSIGNALIZED_NODE_TABLE Test_Unsignalized_Node_Table

# The signalized node table name.
NET_SIGNALIZED_NODE_TABLE Test_Signalized_Node_Table

# The phasing plan table name.
NET_PHASING_PLAN_TABLE Test_Phasing_Plan_Table

# The timing plan table name.
NET_TIMING_PLAN_TABLE Test_Timing_Plan_Table

# The speed table name.
NET_SPEED_TABLE Test_Speed_Table

# The lane use table name.
NET_LANE_USE_TABLE Test_Lane_Use_Table

# The transit stop table name.
NET_TRANSIT_STOP_TABLE Test_Transit_Stop_Table

# The signal coordinator table name.
NET_SIGNAL_COORDINATOR_TABLE Test_Signal_Coordinator_Table

# The detector table name.
NET_DETECTOR_TABLE Test_Detector_Table

# The turn prohibition table name.
NET_TURN_PROHIBITION_TABLE Test_Turn_Prohibition_Table

# The barrier table name.
NET_BARRIER_TABLE Test_Barrier_Table

# The activity location table name.
NET_ACTIVITY_LOCATION_TABLE Test_Activity_Location_Table

# The process link table name.
NET_PROCESS_LINK_TABLE Test_Process_Link_Table

# The study area links table name.
NET_STUDY_AREA_LINKS_TABLE Test_Study_Area_Link_Table
```

Table 58 through Table 75 list the contents of the tables.

Table 58. Test node table.

ID	EASTING	NORTHING	ELEVATION	NOTES
8520	3000	2500	1000	
8521	2000	1500	1000	
14136	3000	1500	1000	
14141	3000	4000	1000	
14142	3000	5000	1000	
14340	4000	4000	1000	
8525	3000	500	1000	
8522	2000	4000	1000	
8523	1000	1500	1000	
8524	2000	500	1000	
8606	500	500	900	
8603	4000	500	1000	
8608	4000	5000	1000	
8600	500	4000	750	
8610	500	5000	1000	

Table 59. Test link table.

ID	NAME	NODEA	NODEB	PERMLANESA	PERMLANESB	LEFTPCKTSA	LEFTPCKTSB	RGHTPCKTSA	RGHTPCKTSB	TWOWAYTURN	LENGTH	GRADE	SETBACKA	SETBACKB	CAPACITYA	CAPACITYB	SPEEDLMTA	SPEEDLMTB	FREESPDA	FREESPDB	FUNCTCLASS	THRUA	THRUB	COLOR	VEHICLE	NOTES
9704	2nd Street	8521	8523	2	2	0	0	0	0	F	1000	0	3	3	800	1000	20	20	25	25	OTHER	12407	9704	4	AUTO	
9705	Avenue B	8521	8522	1	1	0	0	0	0	F	2500	0	6	12	800	1000	20	20	25	25	ZONECONN	9706	11487	1	AUTO	
9706	Avenue B	8521	8524	1	1	0	0	0	0	F	1000	0	6	6	800	1000	20	20	25	25	RAMP	9705	2758	3	AUTO	
11486	Avenue C	14141	14142	3	3	0	0	0	0	F	1000	0	13.5	6	800	1000	20	20	25	25	FRONTAGE	28800	2752	4	AUTO	
11487	3rd Street	8522	14141	3	6	0	0	0	0	F	1000	0	3	9	800	1000	20	20	25	25	SECARTER	2754	11495	2	AUTO	
11495	3rd Street	14141	14340	6	3	0	0	0	0	F	1000	0	18	6	800	1000	20	20	25	25	COLLECTOR	11487	2750	3	AUTO	
12384	Avenue C	14136	8520	4	4	0	1	0	1	T	1000	0	6	0	800	1000	20	20	25	25	FREEWAY	28804	28800	4	AUTO	
12407	2nd Street	8521	14136	2	2	0	0	1	0	F	1000	0	3	12	500	500	20	20	25	25	XPRESSWAY	9704	12384	2	AUTO	
28800	Avenue C	8520	14141	3	4	0	1	0	1	T	1500	0	0	13.5	800	1000	20	20	25	25	PRIARTER	12384	11486	1	AUTO	
28804	Avenue C	14136	8525	5	4	0	0	0	0	F	1000	0	6	6	800	1000	20	20	25	25	LOCAL	12384	2759	1	AUTO	
2759	1st Street	8525	8603	2	3	0	0	0	0	F	1000	0	18	6	800	1000	20	20	25	25	LOCAL	2758	2750	2	AUTO/BUS	
2750	Avenue D	8603	14340	2	3	0	0	0	0	F	3500	0	6	13.5	800	1000	20	20	25	25	LOCAL	2759	2751	4	AUTO/BUS	
2751	Avenue D	14340	8608	3	2	0	0	0	0	F	1000	0	13.5	6	800	1000	20	20	25	25	LOCAL	2750	2752	1	AUTO/BUS	
2752	4th Street	8608	14142	2	2	0	0	0	0	F	1000	0	6	9	800	1000	20	20	25	25	LOCAL	2751	2753	2	AUTO/BUS/ LIGHTRAIL	
2753	4th Street	14142	8610	1	1	0	0	0	0	F	2500	0	9	6	800	1000	20	20	25	25	LIGHTRAIL	2752	2755	3	LIGHTRAIL	
2755	Avenue A	8610	8600	2	2	0	0	0	0	F	1000	-25	3	9	800	1000	20	20	25	25	LOCAL	2753	2756	1	AUTO/LIGHTRAIL	
2754	3rd Street	8600	8522	2	4	0	0	0	0	F	1500	16.7	6	3	800	1000	20	20	25	25	LOCAL	2755	11487	4	AUTO	
2756	Avenue A	8600	8606	3	2	0	0	0	0	F	3500	4.3	9	6	800	1000	20	20	25	25	LOCAL	2755	2757	3	AUTO/LIGHTRAIL	
2757	1st Street	8606	8524	2	2	0	0	0	0	F	1500	6.7	6	3	800	1000	20	20	25	25	LOCAL	2756	2758	2	AUTO/LIGHTRAIL	
2758	1st Street	8524	8525	2	2	0	0	0	0	F	1000	0	3	12	800	1000	20	20	25	25	LOCAL	2757	2759	4	AUTO/BUS/ LIGHTRAIL	

Table 60. Test speed table.

LINK	NODE	SPEEDLMT		VEHICLE	STARTTIME	ENDTIME	NOTES
2758	8524	15	20	BUS	ALL00:00	ALL24:00	
2758	8525	15	18	BUS	ALL00:00	ALL24:00	

Table 61. Test pocket lane table.

ID	NODE	LINK	OFFSET	LANE	STYLE	LENGTH	NOTES
85201	8520	12384	0	1	M	100	
85206	8520	12384	0	6	M	200	
85213	8521	12407	450	3	P	100	
141411	14141	28800	0	1	T	200	
141416	14141	28800	0	6	T	300	

Table 62. Test lane use table.

NODE	LINK	LANE	VEHICLE	RESTRICT		ENDTIME	NOTES
8606	2757	2	AUTO/HOV3	O	ALL00:00	ALL24:00	
8524	2757	1	LIGHTRAIL	R	ALL00:00	ALL24:00	
8524	2758	1	LIGHTRAIL	R	ALL00:00	ALL24:00	
8524	2758	2	AUTO	R	ALL00:00	ALL24:00	
8525	2758	1	AUTO	N	ALL00:00	ALL24:00	
8525	2758	2	LIGHTRAIL	N	ALL00:00	ALL24:00	
8606	2756	1	LIGHTRAIL	R	ALL00:00	ALL24:00	
8600	2756	1	LIGHTRAIL	R	ALL00:00	ALL24:00	
8600	2755	2	LIGHTRAIL	N	ALL00:00	ALL24:00	
8610	2755	1	LIGHTRAIL	R	ALL00:00	ALL24:00	
14142	2752	1	LIGHTRAIL	R	ALL00:00	ALL24:00	
14142	2752	2	AUTO	R	ALL00:00	ALL24:00	
8608	2752	1	AUTO	N	ALL00:00	ALL24:00	
8608	2752	1	LIGHTRAIL	R	ALL00:00	ALL24:00	

Table 63. Test parking table.

ID	NODE	LINK	OFFSET	STYLE	CAPACITY	GENERIC	VEHICLE	STARTTIME	ENDTIME	NOTES
1001	8520	28800	400	LOT	50	T	AUTO	ALL00:00	ALL24:00	
1002	14136	12384	300	PRSTR	10	T	AUTO/TAXI	ALL00:00	ALL24:00	
1003	14136	12407	200	HISTR	10	T	ANY	ALL00:00	ALL24:00	
1004	8521	12407	200	DRVWY	1	F	ANY	ALL00:00	ALL24:00	
1005	8525	2758	370	LOT	1	F	BUS	ALL00:00	ALL24:00	
1006	14142	2752	650	LOT	0	F	ANY	ALL00:00	ALL24:00	

Table 64. Test barrier table.

ID	NODE	LINK		LANE	STYLE	LENGTH	
9001	8600	2756	450	1	BARRIER	200	

Table 65. Test transit stop table.

ID	NAME	NODE	LINK	OFFSET		STYLE	CAPACITY	NOTES
3001	1st & C NE	8525	2759	400	BUS	STOP	2	
3002	1st & C SW	8525	2758	350	BUS/LIGHTRAIL	STATION	0	
3003	1st & B	8524	2757	650	LIGHTRAIL	YARD	0	
3004	4th & A	8610	2755	600	LIGHTRAIL	STOP	2	
3005	4th & C	14142	2752	650	BUS/LIGHTRAIL	STATION	0	
3006	3rd & D	14340	2750	400	BUS	STOP	1	

Table 66. Test lane connectivity table.

NODE	INLINK	INLANE	OUTLINK	OUTLANE	NOTES
14141	11487	1	11486	1	
14141	11487	2	11486	2	
14141	11487	3	11495	1	
14141	11487	4	11495	2	
14141	11487	5	11495	3	
14141	11487	6	28800	3	
14141	11486	1	11495	1	
14141	11486	2	28800	1	
14141	11486	3	28800	2	
14141	11486	3	11487	3	
14141	11495	1	28800	1	
14141	11495	2	28800	2	
14141	11495	3	11487	1	
14141	11495	4	11487	2	
14141	11495	5	11487	3	
14141	11495	6	11486	3	
14141	28800	1	11487	1	
14141	28800	2	11487	2	
14141	28800	3	11486	1	
14141	28800	4	11486	2	
14141	28800	5	11486	3	
14141	28800	6	11495	3	
8520	12384	2	28800	2	
8520	12384	3	28800	3	
8520	12384	4	28800	4	
8520	12384	5	28800	5	
8520	28800	1	12384	1	
8520	28800	2	12384	2	
8520	28800	3	12384	3	
8520	28800	3	12384	4	
14136	12407	1	12384	1	
14136	12407	2	28804	4	
14136	12384	1	28804	1	
14136	12384	2	28804	2	
14136	12384	3	28804	3	
14136	12384	4	28804	4	
14136	12384	4	12407	2	
14136	28804	1	12407	1	
14136	28804	1	12384	2	
14136	28804	2	12384	3	
14136	28804	3	12384	4	
14136	28804	4	12384	5	
14136	28804	5	12384	6	
8521	12407	1	9704	1	
8521	12407	1	9706	1	
8521	12407	2	9704	2	
8521	12407	2	9705	1	
8521	9704	1	12407	1	
8521	9704	1	9705	1	
8521	9704	2	12407	2	
8521	9704	2	9706	1	
8521	9705	1	9706	1	
8521	9705	1	9704	2	
8521	9705	1	12407	1	

NODE	INLINK	INLANE	OUTLINK	OUTLANE	NOTES
8521	9706	1	9705	1	
8521	9706	1	12407	2	
8521	9706	1	9704	1	
14340	2750	1	11495	1	
14340	2750	2	11495	2	
14340	2750	2	11495	3	
14340	2750	3	2751	1	
14340	2750	3	2751	2	
14340	11495	1	2751	1	
14340	11495	2	2751	2	
14340	11495	2	2750	1	
14340	11495	3	2750	2	
14340	2751	1	2750	1	
14340	2751	1	11495	4	
14340	2751	2	2750	2	
14340	2751	2	11495	5	
14340	2751	3	11495	6	
8608	2751	1	2752	1	
8608	2751	2	2752	2	
8608	2752	1	2751	1	
8608	2752	1	2751	2	
8608	2752	2	2751	3	
8603	2759	1	2750	1	
8603	2759	2	2750	2	
8603	2759	3	2750	3	
8603	2750	1	2759	1	
8603	2750	2	2759	2	
8606	2757	1	2756	1	
8606	2757	1	2756	2	
8606	2757	2	2756	3	
8606	2756	1	2757	1	
8606	2756	2	2757	2	
8610	2753	1	2755	1	
8610	2755	1	2753	1	
8600	2756	1	2755	1	
8600	2756	2	2755	2	
8600	2756	2	2754	3	
8600	2756	3	2754	4	
8600	2754	1	2756	1	
8600	2754	2	2755	2	
8600	2755	1	2756	1	
8600	2755	2	2756	2	
8600	2755	1	2754	1	
14142	2753	1	2752	1	
14142	11486	3	2752	2	
14142	2752	1	2753	1	
14142	2752	2	11486	1	
8522	2754	1	11487	2	
8522	2754	2	11487	3	
8522	2754	3	11487	4	
8522	2754	4	11487	5	
8522	2754	4	9705	1	
8522	9705	1	11487	6	
8522	11487	1	9705	1	
8522	11487	2	2754	1	
8522	11487	3	2754	2	
8524	2758	1	2757	1	

NODE	INLINK	INLANE	OUTLINK	OUTLANE	NOTES
8524	2758	2	2757	2	
8524	2758	2	9706	1	
8524	9706	1	2757	2	
8524	9706	1	2758	2	
8524	2757	1	2758	1	
8524	2757	2	2758	2	
8524	2757	1	9706	1	
8525	2758	1	2759	2	
8525	2758	2	2759	3	
8525	2759	1	2758	1	
8525	2759	2	2758	2	
8525	2759	2	28804	5	
8525	2758	2	28804	1	
8525	28804	1	2759	1	
8525	28804	2	2759	2	
8525	28804	3	2759	3	
8525	28804	4	2758	2	

Table 67. Test unsignalized node table.

NODE		SIGN	NOTES
8520	12384	Y	
8520	28800	N	
14136	12407	S	
14136	12384	N	
14136	28804	N	
8610	2753	N	
8610	2755	N	
14142	2753	N	
14142	11486	S	
14142	2752	N	
8608	2751	N	
8608	2752	N	
8600	2756	N	
8600	2754	S	
8600	2755	N	
8522	2754	N	
8522	9705	S	
8522	11487	N	
14340	2751	S	
14340	11495	S	
14340	2750	S	
8606	2756	N	
8606	2757	N	
8524	2757	N	
8524	2758	N	
8524	9706	Y	
8525	2758	N	
8525	2759	N	
8525	28804	S	
8603	2759	N	
8603	2750	N	

Table 68. Test signalized node table.

NODE	TYPE	PLAN	OFFSET	STARTTIME		RING	ALGORITHM	NOTES
14141	T	1	19	ALL00:00	0	S	B	
8521	A	2	0	ALL18:00	0	S	B	
8521	A	3	0	WKD07:00	0	S	B	

Table 69. Test phasing plan table.

NODE	PLAN	PHASE	INLINK	OUTLINK	PROTECTION	DETECTORS	NOTES
14141	1	1	11487	11495	U	0	
14141	1	1	11487	28800	P	0	
14141	1	1	11495	11487	U	0	
14141	1	1	11495	11486	P	0	
14141	1	1	11486	11487	S	0	
14141	1	1	28800	11495	S	0	
14141	1	2	11487	28800	P	0	
14141	1	2	11495	11486	P	0	
14141	1	2	11486	11495	P	0	
14141	1	2	28800	11487	P	0	
14141	1	2	28800	11495	S	0	
14141	1	2	11486	11487	S	0	
14141	1	3	11487	28800	P	0	
14141	1	3	28800	11487	P	0	
14141	1	3	28800	11486	U	0	
14141	1	3	28800	11495	P	0	
14141	1	3	11495	11486	S	0	
14141	1	3	11486	11487	S	0	
14141	1	4	11487	28800	P	0	
14141	1	4	11486	11495	U	0	
14141	1	4	11486	28800	U	0	
14141	1	4	11486	11487	P	0	
14141	1	4	28800	11486	U	0	
14141	1	4	28800	11495	P	0	
14141	1	4	11495	11486	S	0	
14141	1	5	11487	11486	P	0	
14141	1	5	11487	28800	P	0	
14141	1	5	11495	28800	P	0	
14141	1	5	11495	11486	S	0	
14141	1	5	11486	11487	P	0	
14141	1	5	28800	11495	P	0	
14141	1	6	11487	28800	P	0	
14141	1	6	11495	28800	P	0	
14141	1	6	11495	11487	U	0	
14141	1	6	11495	11486	P	0	
14141	1	6	11486	11487	S	0	
14141	1	6	28800	11495	P	0	
8521	2	1	9705	9704	U	6006	
8521	2	1	9705	9706	U	6006	
8521	2	1	9705	12407	U	6006	
8521	2	1	9706	9705	U	6003	
8521	2	1	9706	12407	U	6003	

NODE	PLAN	PHASE	INLINK	OUTLINK	PROTECTION	DETECTORS	NOTES
8521	2	1	9706	9704	U	6003	
8521	2	2	12407	9704	U	6004/6005	
8521	2	2	12407	9705	U	6005	
8521	2	2	12407	9706	U	6004	
8521	2	2	9704	12407	U	6001/6002	
8521	2	2	9704	9705	U	6001	
8521	2	2	9704	9706	U	6002	
8521	3	1	9705	9704	U	6006	
8521	3	1	9705	9706	U	6006	
8521	3	1	9705	12407	U	6006	
8521	3	1	9706	9705	U	6003	
8521	3	1	9706	12407	U	6003	
8521	3	1	9706	9704	U	6003	
8521	3	2	12407	9706	P	6004	
8521	3	2	29704	9705	P	6001	
8521	3	3	12407	9704	U	6004/6005	
8521	3	3	12407	9705	U	6005	
8521	3	3	12407	9706	U	6004	
8521	3	3	9704	12407	U	6001/6002	
8521	3	3	9704	9705	U	6001	
8521	3	3	9704	9706	U	6002	

Table 70. Test timing plan table.

PLAN	PHASE	NEXT-PHASES	GREENMIN	GREENMAX	GREENEXT	YELLOW	REDCLEAR	GROUPFIRST	NOTES
1	1	2	35	0	0	4	0	1	
1	2	3	5	0	0	3	0	0	
1	3	4	8	0	0	3	0	0	
1	4	5	32	0	0	4	0	0	
1	5	6	9	0	0	3	0	0	
1	6	1	1	0	0	3	0	0	
2	1	2	12	30	4	3	0	1	
2	2	1	10	40	4	3	0	0	
3	1	2	12	30	4	3	1	1	
3	2	3	4	8	2	3	0	0	
3	3	1	10	20	4	3	1	0	

Table 71. Test detector table.

ID	NODE	LINK	OFFSET	LANEBEGIN	LANEEND	LENGTH	STYLE	COORDINATR	CATEGORY	NOTES
5001	14142	2753	350	1	1	3	PASSAGE	1000	PERFECT	
5002	14142	11486	250	1	3	3	PRESENCE	1000	PERFECT	
5005	14142	2752	300	1	2	3	PASSAGE	1000	A	
6001	8521	9704	3	1	1	100	PRESENCE	8521	PERFECT	
6002	8521	9704	3	2	2	100	PRESENCE	8521	PERFECT	
6003	8521	9706	6	1	1	100	PRESENCE	8521	PERFECT	
6004	8521	12407	3	1	1	100	PRESENCE	8521	PERFECT	
6005	8521	12407	3	2	2	100	PRESENCE	8521	PERFECT	
6006	8521	9705	6	1	1	100	PRESENCE	8521	PERFECT	

Table 72. Test signal coordinator table.

ID	TYPE	ALGORITHM	NOTES
1000			

Table 73. Test activity location table.

ID	NODE	LINK	OFFSET	LAYER		NORTHING	ELEVATION	ACCESS	HOME	WORK	NOTES
23	8524	9706	200	AUTO	2000	700	1000	0.00	1.0	0.0	
24	8521	12407	300	BUS	2300	1500	1000	375.	0.0	1.0	

Table 74. Test process link table.

ID	FROMID	FROMTYPE	TOID	TOTYPE	DELAY	COST	NOTES
123	3003	TRANSIT	23	ACTIVITY	10	20	
124	24	ACTIVITY	1003	PARKING	30	40	

Table 75. Test study area link table.

ID	BUFFER	NOTES
9704	N	
9705	N	
9706	N	
11486	N	
11487	N	
11495	N	
12384	N	
12407	N	
28800	N	
28804	N	
2759	Y	
2750	Y	
2751	Y	
2752	Y	
2753	Y	
2755	Y	
2754	Y	
2756	Y	
2757	Y	
2758	Y	

Appendix B: Vehicle Files Examples

Example 1:

Household 1460 has two vehicles (500100 and 500101); both start at the home location (78) and are of network type auto (1).

The user-defined emissions vehicle subtype (10) is the same for both vehicles. The optional user-defined integer field is present in this file. One integer field is an indicator of the maintenance level of the vehicle. Note that the second vehicle (500101) has unknown/unused value (-1) for the first optional integer field.

```
HHID  VEHICLE  LOCATION  VEHTYPE  VSUBTYPE
1460   500100    78        1         10        30
1460  500101    8        1        10       -1
```

Example 2:

Read all of the data in the vehicle file then write the vehicle information to another file. The data for each vehicle is stored in a `VehicleData` structure.

```
#include <stdio.h>
#include <vehio.h>

int main(int argc, char *argv[])
{
   FILE   *fp;
   FILE   *outfp;
   int    count = 0;
   const VehicleData       *veh;
   Tveh DataHeader         *header;

   if (argc < 3) {
      fprintf(stdout, "Usage:  testveh <veh input file> <output file>\n");
      exit(0);
   }

   fp = fopen(argv[1], "r");
   if (fp == NULL) {
      printf("Failed to open file %s...exiting\n", argv[1]);
      exit(0);
   }

   outfp = fopen(argv[2], "w");
   if (outfp == NULL) {
      printf("Failed to open file %s...exiting\n", argv[2]);
      exit(0);
   }

VehDataReadHeader (fp, header); vehDataWriterHeader (out fp; header);
   while (moreVehicles(fp)) {
      veh = getNextVehicle(fp);
      if (veh == NULL) {
         fprintf(stderr, "Error FAILED to get vehicle...exiting\n");
         break;
```

```
        }
        count++;
        if (!writeVehicle(outfp, veh)) {
            fprintf(stderr, "Failed to write vehicle %d\n", veh>fVehicleId);
        }
    }

    fclose(fp);
    fclose (outfp);
    return 0;
}
```

Appendix C: Vehicle Prototype Files Examples

VEHTYPE	VSUBTYPE	MAXVEL	MAXACCEL	LENGTH	CAPACITY
1	1	44.0	4.5	5.5	2
1	2	40.7	3.0	4.4	5
1	3	37.5	2.2	9.8	8
2	0	39.5	1.8	12.5	2
5	0	30.	1.1	15.0	52

Appendix D: Network Error Codes

Error codes for the Network representation are in the range 19000 – 19999.

Table 76. Network error codes.

Code	Description
19001	Caught signal.
19002	Assertion failed.
19003	Invalid program arguments.
19004	Standard exception.
19005	Unknown exception.
19006	Network subsystem problem.
19007	Not found.
19008	Illegal value.
19009	Undefined control.
19010	Cannot read.
19011	Input or output error.
19012	Unrecognized functional class.
19013	Movement not assigned to a phase.
19014	Lanes not assigned to a phase.
19015	Incoming degree does not match outgoing degree.
19016	Multiple functional classes.
19017	Respecifying straight link.
19018	Incoming degree greater than 4.
19019	Null control assigned.
19020	OTHER functional class.
19021	LIGHTRAIL functional class.

Appendix E: Transit Error Codes

Error codes for Transit are in the range 30000 – 30999.

Table 77. Transit error codes.

Code	Description
30001	Caught signal.
30002	Assertion failed.
30003	No transit route file specified.
30004	Cannot open transit route file.
30005	No transit schedule file specified.
30006	Cannot open transit schedule file.
30007	No transit zone file specified.
30008	Cannot open transit zone file.
30009	Illegal value in field `TRANSIT_TYPE` of transit zone table.

Appendix F: Vehicle Error Codes

Error codes for Vehicle are in the range 33000 – 33999.

Table 78. Vehicle error codes.

Code	Description
33001	Caught signal.
33002	Assertion failed.
33003	Exception has occurred.
33004	Network exception has occurred.
33005	Unknown exception has occurred.
33006	Invalid program usage.
33007	Failed to open file for reading.
33008	Failed to open file for writing.
33009	Failed to write record to file.
33010	Mandatory file not specified.
33011	Failed to construct network.
33012	Person demographic header not specified.
33013	Specified person demographic not found in population.
33014	Home activity location not found in network activity location table.
33015	Failed to read file header line(s).
33016	No parking location accessible via process links from specified activity location.

Volume Two: Index